REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
07-08-2006

2. REPORT TYPE
Final Report

3. DATES COVERED *(From – To)*
13 June 2003 - 13-Jun-06

4. TITLE AND SUBTITLE
Adaptive Associative Scale-Free Maps for Fusing Human and Robotic Intelligence

5a. CONTRACT NUMBER
FA8655-03-1-3036

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Professor Andras Lorincz

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Eotvos Lorand University
Pazmany Peter setany 1/C
Budapest H-1117
Hungary

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
EOARD
PSC 821 BOX 14
FPO 09421-0014

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Grant 03-3036

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report results from a contract tasking Eotvos Lorand University as follows: The Grantee will perform research in high level Information Fusion focused on real-time management and cooperative planning in supervised autonomous systems. The research will be guided by two principal considerations: A) human knowledge should influence and guide information gathering, and B) collected, machine-processed information should reflect human intelligence and be readily comprehensible to the user. The domain is web-based knowledge extraction using crawlers, word-maps, and scale-free network models. Complete details of expected results and technical contributions are provided in the technical proposal.

15. SUBJECT TERMS
EOARD, Scale Free Small Worlds, Information Fusion

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18, NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			PAUL LOSIEWICZ, Ph. D.
UNCLAS	UNCLAS	UNCLAS	UL	66	19b. TELEPHONE NUMBER *(Include area code)* +44 20 7514 4474

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39-18

Adaptive Associative Scale-Free Maps for Fusing Human and Robotic Intelligences

Final Report
Contract: FA8655-03-1-3036
July 2003 – June 2006

Contact
István Alföldi
John von Neumann Computer Society (NJSzT)

Principal Investigator
András Lőrincz
Eötvös Loránd University

Budapest

June 2006

Contents

List of Figures

Chapter 1

Executive Summary

1.1 Introduction

The nature of intelligence has been a long-standing mystery for scientific research. Until it will be solved, machine intelligence remains undefined, some may say meaningless. Thus, we are forced to come up with a hypothesis about intelligence to enable us to talk about the integration of human and machine intelligences subject to the constraint/context of our hypothesis.

In our view, intelligence is mastering to deal with and to resolve a special class of combinatorial problems; those which are hard to solve but are easy to verify [Lőrincz, 2004]. An important assumption of this definition is the existence of 'components', the parts that information is made of. Combinatorics then concerns components of the available information in two ways (i) the combination of components into new ones and (ii) the deciphering of combined components.

Now we can phrase what we mean by the task of fusing intelligences:

Point 1 Independent searches for components

Point 2 Decision – in agreement with other agents – about the symbols (e.g., the words) that represent the components

Point 3 Jointly accepted methods of recognition by means of the agreed components

Point 4 The list of known combinations of components, inferences by means of components, inferences about the appearance of novel components or the appearance of novel combinations of components

1.2 Solution

Our project AFRL IRI FA8655-03-1-3036 is concerned with **Point 2** above, with two agents; the human user and the machine. An interaction scheme was designed. The scheme required particular items that we had to develop and study, including special learning schemes for internet crawlers [Palotai et al., 2006b], synonym generation and synonym based explanation system [Gábor et al., 2005], map based internet navigation and map based explanation. We have found intriguing results concerning learning in scale-free small worlds (SFSW) [Palotai et al., 2005a, Palotai et al., 2006a]. SF-SWs are very important, because they seem to form in every evolutionary system, including – among others – the Internet and the language.

During our developments we have found that one of the most serious bottlenecks of the research is the verification (evaluation) of the methods. At start, we intended to rely on human experts. However, this idea had to be rejected, because we were unable to design an objective measure for judging the human expert and his/her activities when interacting with the machine. After all, the project concerns an algorithm, which is supposed to pass the Turing test [Lőrincz and Szirtes, 2003] in the long run, but there is no general performance measure for this test.

During the project, we conducted several experiments to solve these problems. We have tried:

1. *crawler competition* over the Internet,
2. crawler competition on a downloaded and *modifiable portion of the Internet*,
3. *crawler communication* in a downloaded and modifiable portion of the Internet,
4. *human competition* aided by machine intelligence, and
5. *student examination* to test overview of course material.

Eventually have figured out a method to fuse intelligences. In essence, the solution has intermingled steps: (i) an unknown topic is created by means of a decision surface in document space and humans try to find the components of the topic: they try to identify the topic by creating word maps. Performance of the word maps is judged by Internet crawlers that crawl with the word maps, namely, according to the *precision* and the *recall* rates of the crawlers. These numbers are computed for the downloaded documents by means of the decision surface. Humans are credited according to the quality of their crawler. This is our solution for **Point 2** if it is under human supervision.

The solution has promises for **Points 1** and **Point 3**, too: strategies used by humans were identified by interviewing them. These are ingenious algorithms that can be applied separately or in combination. The loop can be closed if the crawlers use these human strategies. Then, in the future, crawlers themselves may search for novel components and the word map based identification of the topic will become a *joint effort*.

This route does not offer a straightforward solution for **Point 4** and this point remains open. We *believe* that *belief networks* may be able to tackle this last point. The advance of fast belief propagation algorithms [Hinton et al., 2006], the highly clustered nature of SFSWs, and existence of fast and local distributed clustering algorithms in SFSWs (see, e.g., [Lőrincz et al., 2004b] and references therein) have the promise of fast machine inferencing in human-computer collaboration.

Our results, which are detailed in this *Final Report*, show the promise of the approach.

Part I

Overview of Research and Achievements

Chapter 2

Motivation

2.1 What can we learn from psychology?

Here, we shall review two theories on human cognition. On the basis of certain psychophysical experimental data, Biederman suggests that recognition occurs by the recognition of the components [Biederman, 1987]. This opinion will be called factorial theory. A typical example is the recognition of faces by means of the components, e.g., eyes, nose, mouth and their relative arrangements. The promise of this theory is that it can be generalized to spatio-temporal patterns. Then, the recognition by components *and* pattern completion may also concern (i) prediction of future events and/or (ii) filling in missing past events.

The other theory claims that cognition is categorization (see, e.g., [Harnad, 2003] and references therein). Categorization here means the forming of decision surfaces between different sample sets. Under this condition, the components of the decision can not be established, learning is hard earned, it proceeds by trial-and-error and the result of learning is not accessible to introspection. Upon learning, discrimination capabilities typically change; discrimination is enhanced between categories and decreases within the categories. The phenomenon is called categorical perception. This view will be referred to as categorization theory. A typical example is the naming of a color.

Both theories are necessary for recognition: we need to recognize the components in order to recognize the whole by means of the components. This 'procedure' is an infinite regress, unless at some point hard earned categories help us. For example, the color of the face or certain components of the mouth, etc., may not have components and their recognition may rely on categorical perception.

2.2 How can we translate these ideas to communication and intelligence?

Communication is about passing information. The passed information then – when used – undergoes verification. First, for the sake of clarity, let us consider evolution. During evolution, information is passed from individuals to individuals by means of the genes, which are the compressed forms of 'success stories' (=: the fittest survive and can multiply). Then, the new generation unfolds these success stories and 'verifies' them. If verification is successful, then the compressed information can be passed to the next generation. In the case of evolution, propagation of information concerns propagation through time.

This situation changes for communication between agents. Now, information transfer occurs in space, too. There are new features in this case, information transfer gains novel dimensions. Transferred information may be about the solution of a given problem or it may be about an observation that decreases limitations of observation of individual agents and helps pattern completion and pattern recognition in various circumstances. In both cases, the new information – if used then it – undergoes verification.

Intriguingly, verification can be related to the theory of computation. From the point of view of communication, there are two basic types of computational tasks determined by the polynomial, non-polynomial but polynomially verifiable (PV) and non-PV classification of computational problems.

We say that a problem is polynomial (it is easy to solve), if it can be solved in polynomial time. On the other hand, a problem is PV (it is easy to verify), if it has solution instances that can be verified in polynomial time. Non-PV problems have instances, which can not be verified in polynomial time (they are hard to verify).

In turn, *from the point of view of communication*, we have two groups: The first group can be called as not worth to communicate type; *non-WTC type* . This type is either easy to solve and easy to verify, or hard to solve and hard to verify. The other type is hard to solve but easy to verify and, in turn, it is of *WTC type*: It is worth to communicate the solution. For non-WTC type problems communication is simply an overhead, it requires computational power and no gains can be expected by communication. On the other hand, WTC type problems – according to theory of computation – may have exponential gains if communication and then verification is possible. As an example, consider the Travelling Salesman problem. The complexity of the problem scales very quickly with the number of cities, but the complexity of verification scales linearly, we just have to 'follow' the solution to verify it.

Chapter 3

Solution

For systems dealing with the semantic content of documents, the short summary of the content, or the context of any document is of critical importance. Comparing documents by content, finding out which documents are relevant for a user can be performed by highlighting important fractions of the document. There are trivial ways to do this and such methods have been included into MS Word since at least 1997. Such simple methods search for frequent words and select parts at the sentence level. This particular method of document compression is not particularly useful if the content and/or the context of the document are unclear for us.

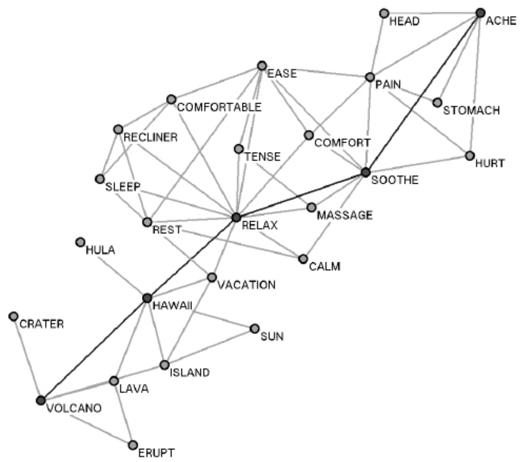


Figure 3.1: **Human association network**
The associative network was created by providing cue words, collecting response words and compressing the network. This network was published in the technical report of [Steyvers and Tenenbaum, 2005].

Another straightforward way to summarize the content of a document is to extract phrases that have a key role in determining the meaning of a document. There have been promising attempts for automatically extracting keyphrases from documents [Turney, 1999, Frank et al., 1999]. A simple, possibly ordered list of keyphrases conveys many information for human readers. Humans are very good at augmenting bits and pieces of information with background knowledge, like finding out the connections between keyphrases, which greatly helps them in guessing the meaning of a document provided the list of keyphrases. However, this method may be insufficient for an automated system or also for us in crucial cases. Some kind of interaction with the computer may be necessary. For example, when searching for documents in Google, the search engine makes suggestions, whether we would like (i) to sharpen the search by particular logical constructs between keywords or (ii) to find 'similar documents'. In many cases, these options still do not warrant the proper cognitive embedding, i.e., the search engine makes no effort to match our 'thinking'.

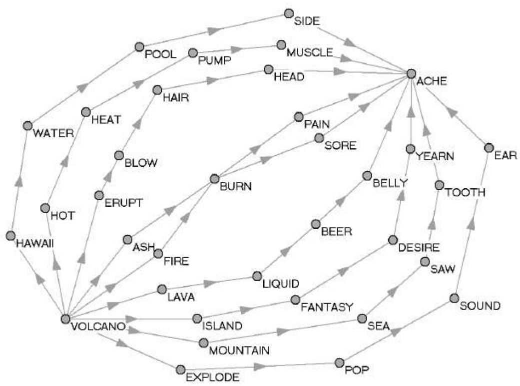


Figure 3.2: **Step-by-step collection of human association network [Steyvers and Tenenbaum, 2005]**
The associative network was created by providing cue words and collecting response words. The network shows different contexts and switching (routes) between those contexts.

In our approach, we rely on maps. This is due to the following:

Human thinking: We are good in associations and associations are map like structures (see, Fig. 3.1 and also Fig. 3.2).

Scale free small worlds: Maps are particularly useful for scale-free small worlds (SFSW), because SFSW structure supports hierarchical clustering.

Natural language: The word associations graph of human languages exhibits the SFSW property, too [Ferrer i Cancho and Sole, 2001, Steyvers and Tenenbaum, 2005]. This underlines the importance of graphs.

The interaction with the human user is depicted in Fig. 3.3

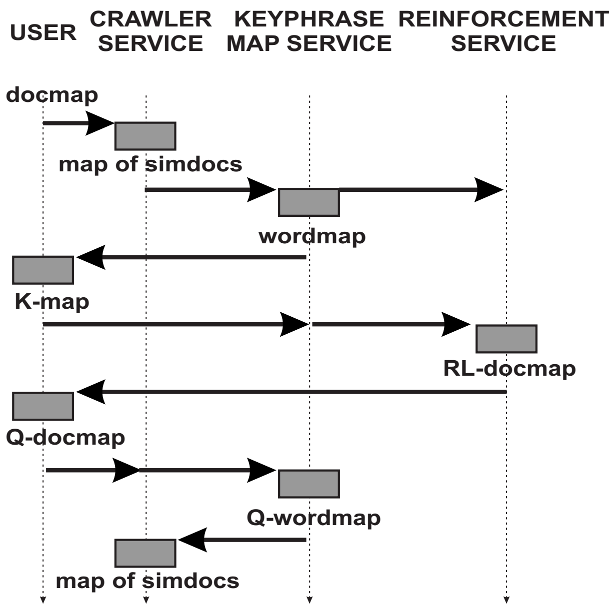


Figure 3.3: **Planned user-computer interaction framework**
Planned framework makes use of keywords and keyword maps. Such maps can be provided and modified by the user, and are also extracted, computed, and compared by the computer for any given document. For more details of the architecture, see our Second Report. Notations: $Q \Rightarrow$ quality, it is achieved *after* user interaction, $K \Rightarrow$ keyword, RL$\Rightarrow$ reinforcement learning, 'sim': similar

3.1 Fusing intelligences: Algorithms

We have developed and implemented algorithms as required by Fig. 3.3. Central pieces of the algorithms are as follows:

1. Distributed bottom-up clustering algorithm that serves to partition large graphs [Lőrincz et al., 2004b].

2. User identification and link highlighting algorithm that assist Internet searches on-line [Palotai et al., 2005b].

3. Selective Weblog algorithm that efficiently partitions scale-free small worlds [Palotai et al., 2006b].

4. Neural network algorithm that can be used for synonym queries. It features pattern completion and can be adapted to the actual topics or database [Gábor et al., 2005].

3.2 Fusing intelligences: Preliminary experiments

We have designed and applied different preliminary experiments before deciding how to combine algorithmic components and how to evaluate the performance of the tool.

Crawler Competition over the Internet: We populated a huge scale-free portion of Internet environment with news foragers. They evolved by a simple internal selective algorithm: selection concerned the memory components, being finite in size and containing the list of most promising supplies. Foragers received reward for locating not yet found news and crawled by using value estimation. Foragers were allowed to multiply if they passed a given productivity threshold. A particular property of this community is that there is no direct interaction (here, communication) amongst foragers that allowed us to study compartmentalization, assumed to be important for scalability, in a very clear form. Experiments were conducted with our novel scalable A-life architecture. These experiments had two particular features. The first feature concerned the environment: a scale-free world was studied as the space of evolutionary algorithms. The choice of this environment was due to its generality in mother nature. The other feature of the experiments concerned the fitness. Fitness was not predetermined by us, but it was implicitly determined by the unknown, unpredictable environment that *sustained* the community and by the evolution of the competitive individuals. We found that the A-life community achieved fast compartmentalization [Palotai et al., 2006b].

Crawler Competition on a downloaded portion of the Internet: The 'No Free Lunch Theorem' claims that for the set of all problems no algorithm is better than random search. Thus, selection can be advantageous only on a limited set of problems. We investigated how the topological structure of the environment influences algorithmic efficiency. We have studied random, scale-free, and scale-free small world (SFSW) topologies. Selective learning, reinforcement learning and their combinations were tried. We found that selective learning is the best in SFSW topology. In non-small world topologies, however, selection looses against the combined algorithm. Learning agents were web-crawlers searching for novel, not-yet-found information. Experiments were run on a large news site and on its downloaded portion. Controlled experiments were performed on this downloaded portion: we modified the topology, but kept the occurrence times of the news. Our findings may have implications for the theory of evolution [Palotai et al., 2005a, Palotai et al., 2006a].

Crawler Communication in a downloaded portion of the Internet: We studied the influence of the topology of the environment on the communication efficiency of crawlers in quest of novel information in different topics on the Internet. They employed a selection based 'Weblog' algorithm, a function approximation based reinforcement learning (RL) algorithm, or their combination. Real data were collected from the Web and scale-free worlds, scale-free small world (SFSW), and random environments were created by reorganizing the links. The Weblog algorithm, in effect, modified the starting URL lists of the crawlers, whilst RL altered the URL orderings. The following predictions were tested: (i) if the communication parameters have to be learned, then the effect of communication is negligible, (ii) whereas if they are aptly fixed, then the communication plays a major role in the system. We demonstrated that in the SFSW environment, the number of relevant documents that had been found was approximately the same regardless of whether the crawlers learned the parameters or whether the parameters were fixed in an appropriate manner. Should the crawlers have learned the parameters of their own topics and transmitted them to the other crawlers, then the number of relevant documents communicated by the RL crawlers was significant. In case of properly fixed parameters the reinforcement learning and the Weblog update algorithm assisted each other independently of the type of the environment. The age of a document depended considerably upon the type of the environment provided that the parameters were fixed suitably. The freshness of a document, i.e., whether it was new or not, however, did not depend on the environment [Lázár et al., 2006].

Human Competition aided by machine intelligence: We had downloaded a large portion of the Internet and stored it locally. We have provided an intermixed set of good and false keywords for some particular *valuable* documents of the downloaded portion of the Internet database. The database, in effect had hidden these valuable documents. The task was to find the documents by means of crawling over the downloaded part of the Internet, using keyphrase extraction, creating keyphrase maps, using these maps for crawling, making queries for Google Desktop by means of the keywords, improving the set of keywords and the keyword maps. The hint that we provided to help searches was a similarity measure between the actual document and the set of valuable documents. This measure was computed automatically if it was requested by the user. The similarity comparisons were made by means of the keyword maps. The user who found the largest number of valuable documents was announced as the winner of the competition. Asking about the experiences of the users, **we found** that keyword maps are better if words are arranged in sets: the sets can be connected by 'OR' logical operation and then the words within sets should be connected by 'AND' operation. In this description contexts are poorly defined, but visualization is eased. The extension to context maps is, however, desirable.

Student examination to test overview of course material: The same method was used as the entrance exam for University students. The task was to identify the good portion of the keywords and to find documents related to those sets. Sets were selected carefully, they required the knowledge from more than one chapter of the material of the course. Based on the exams, we have concluded the following:

1. Indeed, this examination is good to gain an overall picture about the understanding of the material, the missing parts, and it is superior to subjective evaluations.
2. Such evaluations need special permission procedures, because one could design quests that uncover emotional and cognitive 'parameters' of the user.

3.3 Fusing intelligences: The experiment

Several preliminary tests were run in order to debug the system and to add functionalities suggested by the users. In order to have a measure for the success of the search, we have designed a reinforcing agent. The agent was made of a decision surface, a support vector machine (SVM) that was trained on PDF files with a few positive and a large number of negative examples. The positive examples defined the *'search topics'* and if the output of the SVM was 1 (-1) then positive (zero) reinforcement was given to the human user.

The human user could create keyword maps for crawlers and the crawlers crawled the web for documents. The crawl was adapted according to the success defined by the keyword maps: if the keyword map of the document was similar to that of the crawler then the crawler was reinforced and the crawling parameters (but not the keyword map) were changed accordingly [Palotai et al., 2006b]. In effect, the human user was successful if it translated its introspection into keyword maps made of components in a way that crawlers became successful for finding documents over the Internet.

That is, the hidden category defined by the decision surface was broken into components by the human user, who optimized the efficiency of the Internet robot and improved the component description of the subject. In other words, the implicit knowledge hidden by the decision surface was approximately identified by human understandable word clusters in an interaction between the human user and the fleet of Internet crawlers.

Two search topics were used, they are detailed in the rest of this report. An illustration of the results follows below.

Chapter 4

Results and Discussion

The interaction between human and computer was very efficient:

- Humans found a large portion of the documents that were used as positive examples in the SVM procedure to create a decision surface.

- Humans found many other documents that passed the SVM classification.

- Evaluations based on the keyword maps over a portion of the positive examples and a 5 times larger set of data (i) form the neighborhood of these positive examples and (ii) on similar topics, showed that the maps could discriminate between the documents with high confidence.

- Special components of the keyword maps provided extra information about the topics, because users could create disjunctive normal forms for the description of the topics. In effect, they created disjunctions of special conjunctive descriptions by means of the keywords that we extracted for them. Many of the successful conjunctions need explanations: it should be understood why and how the performance was so good over the Internet and if this good performance is changing by time or not. This is where novel knowledge can be created, the very point of fusing intelligences.

Details on the *precision of the keyword map based searches* in terms of the so called *accuracy* and *recall* values are provided in Section 5.5.

Here is an illustration. One of the two topics is 'cortical neural prosthetics'. A number of pdf documents (47) was downloaded from the site of the NINDS (http:// www.ninds.nih.gov/ funding/ research/ npp/ index.htm) from a larger set (about 310 documents) of documents on the more general topic of neural prosthetics.

One of the keyword maps created by one of the users is shown in Fig. 4.1. The clusters represent words in 'AND' relations. One might say that all

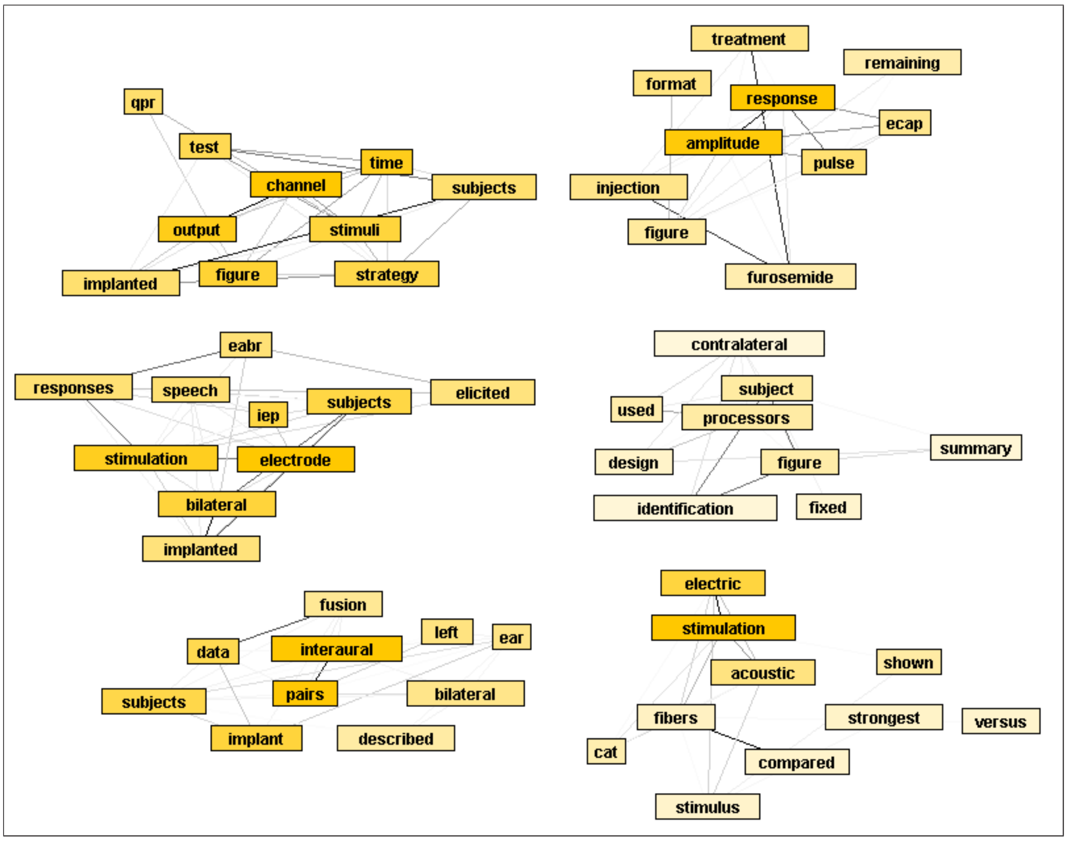


Figure 4.1: **Keyword map for cortical neural prosthetics**

of them but the cluster in the lower right corner is clearly related to implants. The lower right cluster has the following words: *cat electric stimulation acoustic fibers compared shown strongest* and *versus* that seem to have little or no connection to cortical implants.

First, we excluded words *shown strongest* and *versus*. The we tried the rest in Google and received the following hits:

- Comparison of auditory single fiber responses during acoustic and ... Comparison of auditory single fiber responses during acoustic and electric stimulation of the intact cat cochlea. Hartmann R, Topp G, Klinke R.MeSH

- Stochastic properties of cat auditory nerve responses to electric ... Acoustic Stimulation methods Animals Auditory Perception physiology Auditory Threshold physiology Cats Cochlear Nerve physiology Electric Stimulation ...

- Comparison of auditory single fiber responses during acoustic and ...

Comparison of Auditory Single Fiber Responses. During Acoustic and Electric Stimulation. of the Intact Cat Cochlea... R. Hartmann, G. Topp, and R. Klinke

- Auditory nerve fiber responses to electric stimulation: Modulated ... responded to acoustic stimulation or exhibited a multi-modal ... responses of cat auditory nerve fibers to biphasic electrical current pulses,. Ann. Otol.

These hits are clearly connected to the topics. We have deleted words one-by-one, investigated the scripts of the first 50 hits provided by Google, and counted how many of them contained the word implant. For word sets (i) *cat electric stimulation acoustic fibers compared*, (ii) *electric stimulation acoustic fibers compared*, and (iii) *electric stimulation acoustic fibers* we got 5, 7 and 10 hits respectively, and the other hits were also closely related.

Clearly, the combination of the four words *electric stimulation acoustic fibers* has something to do with brain implants. It is hard to see how it may happen, because *acoustic fibers* are cable identifiers for telecommunication engineers and technicians, *electric stimulation* is widely used in therapy, wound care, erotic massagers, and analgesia, expression *electric fibers* are related to piezo-electric tunable optical fibers, and *acoustic stimulation* may concern fluid flow, the fetus, or trapezoid bodies. Still, the combination of the four words is closely related to brain implants, the type of information that might be worth to explore.

A particular part of our methodology is that we have interviewed human participants about their search strategies. Some of the trick they applied can be put into algorithmic form and could be used in the robotic searches. This could greatly improve the efficiency of the machine.

This idea can be taken one step further: the machine could monitor human activity patterns and could mimic and combine those. The machine could also serve the human user by forecasting the next step, e.g., alike in language prediction techniques, e.g., by the compression scheme called 'Prediction by Partial Matching with Information Inheritance' [Shkarin, 2002]. This coder works as follows: Suppose we have processed the first $n-1$ actions $x_1, \ldots, ...x_{n-1}$ of the action stream. Before experiencing the next symbol x_n the machine cam try to guess it, i.e. for every action a the machine estimates the probability $p(a)$ for the event '$x_n = a$'. This probability distribution determines how the next symbol is encoded: The higher $p(a)$, the fewer bits are used for encoding a. If the estimation is good, which means that $p(x_n)$ is high, then a good compression rate is achieved.

4.1 Innovation Engine

Given our results, we can propose an invention engine that combines human intelligence and machine speed. It should look as follows:

Component discovery: This part searches for components, either by human intelligence or by blind methods, see, e.g., [Póczos and Lőrincz, 2005b, Póczos and Lőrincz, 2005a] and references therein. According to our studies, there are component search methods that can break combinatorial explosion by non-combinatorial efforts [Póczos and Lőrincz, 2006].

Recognition by components: This part requires a more sophisticated scheme than we applied here. Our scheme used simple *Disjunctive Normal Forms* (DNF). These forms are general, all logical expression can be translated into DNF. However, they are suboptimal for human intelligence. Human intelligence performs better if sets are grouped and the groups correspond to contexts. This description provides an improved compression: less information is needed within each context. Such graphical representations should be developed and adopted. A good candidate is our bottom-up clustering algorithm [Lőrincz et al., 2004a].

Sample based decisions: Both human and machine intelligence may apply example based classification schemes, the complementer of component based recognition schemes.

Imitation or mimicking: The computer may learn and may predict human action series. Such *macros* could be applied by the computer in parallel, or in combinations. The utilization of such macros is typically implicit and can be made explicit by measuring partial probabilities and by using, e.g., the PPM algorithm.

Components of the maps: Some components of the maps might be very efficient and such knowledge may initiate new questions, novel answers and a better human understanding.

Dialogue system: For convenience, the computer should apply an adaptive conversation system to ease the task for the human user

For more details about the experiments and the experiences, see Part II.

Chapter 5

Conclusions

In summary, great care was needed to design the experiments properly. We found that experiments may need special permissions, because they are able to uncover and quantify emotional characteristics and cognitive capabilities of the users.

Our methodology that can be seen as a learning algorithm that aims to pass the *Turing test*, had considerable success. There are several advantages, such as

- the human friendly identification of decision surfaces in terms of associative maps,
- component formation that enables pattern completion and inferences,
- the symbiosis of crawling and document classification,
- the fresh information provided by the Internet, and
- the potentials of imitation, i.e, the identification of human action series that could be applied by the computer.

There are important tools that could be added. For example, the bottom-up clustering algorithm is to be included for hierarchical graph construction. Other state-of-the-art procedures cost money. Examples include a better interface to Google, an interface to AltaVista NEAR service, which is excellent for finding the most novel synonyms, collecting topic dependent synonym sets, and alike.

The intriguing question for the future is if this architecture could be extended to a datamining *invention engine*. Such extension seems possible for us.

Part II

Results of the last term

5.1 Introduction

Based on the experiences of the test runs performed last term, the concepts and the software tools for examining the interaction between computer and human using keyphrase maps were redesigned.

Last term, preliminary tests were devised and run on a local database downloaded from Wikipedia, a free encyclopedia, to test our concept of an intelligent search engine that combines keyword search engines and personalizable internet crawlers. The role of the keyword search engine was played by Google Desktop, the desktop version of the Google search engine. The search topics were described by a few pages in the database, and the goal was to find certain documents.

Most importantly, tests were now performed on the whole web not on a local database. This enabled us to search in virtually any kind of topics, and made the task more like a real-word problem. Also, the system now supports PDF documents, which contain a great part of the relevant information on today's web. The search topics were also more precisely described by utilizing a document classifier, and the search results were more accurately evaluated by letting the pre-trained classifier decide whether a document is a good one or not.

The goals of the runs were to make the user train good crawlers for a given topic, by developing good keyphrase maps and start URLs for that topic. Crawlers are thought to be good, if they return possibly most of the relevant information, and do not return irrelevant documents. We were also interested in how the keyphrase maps can be used to characterize a certain topic in a human (and also computer) understandable manner, how a topic identified by an document classifier can be covered by keyphrase maps.

5.2 Redesigned software components

5.2.1 Essentials for searching on the web

We have realized that most people who are accustomed to using the Internet for searching documents already have well developed searching habits. The usage of a web browser to look up pages and follow links is essential. Also they are used to view the contents of a document in a structured manner, which is easy to read and quickly examine, like an HTML page. Having this in mind, we have integrated a light-weight browser into the GUI, what is more, this has become the central element. The browser can visualize HTML pages more or less like any other browser, it is possible to follow links, go back to the previous page, or to the next page.

Another tool that people are used to, is keyword search engines. Since we started experimenting with Google, we kept using it in these experiments as well. Google provides a web service to let automated agents perform Google queries. The usage of the Google API requires a license key that can be obtained free of charge, and lets the owner make 1000 queries per day, in order not to flood the server with request. The response of the query is in XML format and contains all the information that one would see when using Google manually. This can be presented to the user in various ways. One response can contain up to 10 results, and the next 10 results can also be requested. The search results (URLs) are presented in a list, and the links can immediately be viewed by selecting them from the list. The details about a result returned by Google can also be observed.

5.2.2 Crawler system

The crawler searching facilities have also been updated, both the crawler engine, and the client interface to it. The crawler engine has been redesigned to become modular and more robust. New features have been added, like suspending and modifying a crawler. Crawlers can be created by sending XML messages to the server describing the crawler. The description of possible XML messages (an XML schema) is generated automatically by the server from the source code. Using this description, the crawlers can also be modified 'on-the-fly'. Which components can be modified, are also generated from source code.

The two main parameters for the crawlers to operate is a keyphrase map (search criterion), and a starting URL. The keyphrase map can be provided via a graphical interface that enables the editing of word graphs. This search criterion can also be modified during the search. During the crawl, the crawlers employ the Weblog algorithm described in an earlier report, which enables them to collect new good starting URLs, by utilizing the reinforcement they receive for returned documents. The Weblog algorithm restarts the crawlers from potentially good starting URLs after a while. A URL is said to be a good starting point, if the discounted cumulative reward collected during a crawl starting from that URL is high. These collected good starting URLs are also returned to the user, who can use this information to start new crawlers.

Crawlers are now enabled to search the web, they operate in compliance with robot standards. Timing constraints are also enforced in order not to flood target servers during the search.

Although the crawler system is implemented in a client-server architecture, presently a local version is used for easier testing. That is, the crawlers of a user run on the same computer, on which he is running the client interface. But the two can be separated, and can also operate in a peer-to-peer fashion using JXTA technology – that we have developed and have tested, but have not tried together with our crawler system yet.

5.2.3 Support for PDF documents

Another important feature we found essential to integrate is the support for parsing PDF documents. Since a huge part of the detailed information that people might be looking for is contained in PDF documents on the web, this feature enables us to perform tests in a 'larger database', using the web more efficiently. Also, the contents of PDF documents are better than that of HTML pages, since PDF documents (like scientific articles) have a better defined topic, while HTML pages may cover more topics in a broader domain. This becomes important when our keyphrase extraction algorithm is applied, since well defined topics are easier to characterize by a few keyphrases.

PDF parsing was made available by utilizing PDFBox[1], a free, open-source PDF library for Java. Support for viewing PDF documents correctly as in commercial viewers was unfortunately not available, instead the text content of the document was extracted, and was visualized as HTML.

5.2.4 Search topic identification

In order to perform large scale tests with the search engine, there is a need for deciding which documents to accept in an automated manner. For this, we trained a Support Vector Machine (SVM) classifier operating on $TF \times IDF$ (Term Frequency $\times$ Inverse Document Frequency) document representations. For each topic used in the tests, a new classifier was trained on about 60 positive and 600 negative samples of scientific articles in PDF format. Each document was transformed into a $TF \times IDF$ representation, reviewed in [Turney, 2002], and an SVM classifier was trained utilizing LibSVM[2], an open source SVM library also available in Java. After this, the classifier was able to predict whether an unseen document is in the topic represented by the positive examples. The classifier was cross validated, and found to have a performance above 95%. This was used to feed back the users whether they have found good documents or not. Partially, the goal of the search was to find many documents in the topic characterized this way.

Users were not only informed about whether the documents found are good or bad, but they were also supplied a score for each document, saying *how good* that document is. This score was calculated by matching the found documents' keyphrase maps to the maps of the positive samples. The maximum of the individual score values to the positive samples were returned to the user. This is also a way to cross-check the keyphrase map based topic identification with the document classification: high score documents should be classified good, while low score documents should be classified out of the

[1] http://www.pdfbox.org

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

topic. This was true in most of the cases, although some outliers were also present, due to the subjective manual selection of the positive samples.

Also, some components in the crawler system need reinforcement from the user to let the reinforcement learning algorithms operate. Unfortunately, human users found it a little uncomfortable to send reinforcement for the crawlers for each document. Thus, we automated the process and utilized the classifier to do it instead of the user: documents classified to be in the topic implied positive reinforcement, while documents outside the topic meant negative reinforcement.

5.3 Initial experiments and experiences

To test the effectiveness of our system, especially whether the keyphrase map concept helps the users in the search, we devised search tasks in scientific topics selected by the test supervisor. The two topics in the first runs were 'Facial expressions' and 'Dyslexia'. The testers were given 5 sample documents in the domain, and were asked to search for similar documents on the web. The testers were members of our group, eight people alltogether.

As described in the previous section, a document classifier was trained on positive and negative samples to be able to classify new documents, whether they belong to the target topic or not. Such way the testers were informed whether they have found a good or a bad document. Along with this information, they were returned a score for each document, indicating how good that document might be. This was based on matching the documents' keyphrase map to that of the sample documents' keyphrase maps, and the maximal match was returned. As natural, the task was to find as much good documents as possible, in a given time interval (typically a few days, taking care of the crawlers for a up to around one hour each day, and letting them run for longer time).

To test whether the keyphrase map concept and the map based document similarity helps users in the search, we divided the testers into two groups: a test group, and a control group. In the test group everything was normal as described above, however, in the control group, the document scores were altered in a way that they conveyed less information about the true value of the document. Since document scores were real values between 0 and 1, 0 indicating a document not in the topic and 1 indicating a perfect match to one of the samples, transforming the scores (with a monotone, nonlinear transformation) so that most of them fell very close to 0.5 seemed to be a reasonable way to make the scores convey minimal information about the documents' content. Of course, the testers did not know about the existance of this transformation, not even about the existence of the two groups. This way we hoped to be able to test them without their knowledge about the tests altering their behavior.

Unfortunately, we were unable to show significant differences in the performances of the two groups, mainly because of three reasons:

- The variation in the strategies and usage preferences of the individuals was so large, that the results were hardly comparable. Some testers used many Google searches, and observed the documents found and sharpened the set of keywords used in the queries, while others started many crawlers after an initial tuning of a keyphrase map (or maps) and let them crawl for a longer period. Some used few crawlers for long runs, some used many crawlers for short runs.
- The testers turned out to be more adaptable than we expected. The ones in the control group with the altered scores realized that the scores had been altered and score distribution has changed (this was because they had previously used the system without the scores being altered) and that there are very little differences between the scores (all of them being close to 0.5). They realized that this ‘new’ scoring system is not so helpful, and the other was easier to use, but they were able to adopt it, and use it like the normal one after a while.
- We have realized, that it is very hard if not impossible to test whether the keyphrase map concept helps people in such a way that they do not know that they are being tested, and a control group also exists. For a true test, it would not be enough to alter the keyphrase map based document scores, but the whole keyphrase map service should be tuned off. But that would mean a complete change in the search tests, and the results would be incomparable. For instance, if keyphrase extraction from documents would not be enabled, people would have to read through possibly a large number of documents, and find the right keywords for themselves.

Testers also realized in this period, that one defect of Google searches can easily be fixed: if one enters a keyword into Google, it will return a huge number of documents containing that keyword, and very few of them will fall into the topic that the user desires. However, if one enters a combination of topic related keywords (like 4-5 words, simply *and*-ed together), then most of the results of the query will be relevant in the topic. Thus, users have developed the following strategy: first, they find good topic related keywords one-by-one, then, they find out how they should be combined, and then they enter many of these combinations to Google, which now returns many relevant documents.

Users have also realized, that the ranking of the documents could still be made better, since Google ranks, documents based on page rank, instead of score. Thus it may happen that many good documents are positioned in the third/fourth/fifth ten of Google responses.

Others, who preferred to use crawlers for the search, also realized that it is easier to cover a topic by many smaller keyphrase maps, than with a large one. This comes from the usage of a broader topic as target. For instance, the topic 'Dyslexia' has many *aspects*. Each aspect must be covered by a smaller map. Then, if we accept any document about dyslexia, we accept a document about *any* of those aspects. Of course, when in a real word search problem, a user would decide which aspects (s)he is interested in (but only after realizing which aspects exist about a topic). Then (s)he would focus on those aspects of interest, this way narrowing the search.

5.4 Modified experiments

Building on the experiences drawn from the previous experiments, the usage of the keyphrase map concept had been rephrased in the next experiments. Since people used many crawlers with smaller maps instead of one larger map, we redesigned the matching of maps in the following way: a word map may contain disconnected components. These components are interpreted as sub-maps of a large map, and the whole is like a disjunction of the parts. On the other hand, the words in a component are treated as *and*-ed together. The maps extracted from documents usually contain one connected component, if not, the words not connected to the main component are usually junk words, and are due to the imperfectness of the keyphrase extraction algorithm. Thus, the matching of the user map to the document maps goes the following way: the components (sub-maps) of the user map are matched one-by-one to the document's map, and the highest match score is returned. This way the user can enumerate the various aspects of a broader topic in one large map by separating it into disconnected components, and the matching accepts any of those aspects.

The idea behind this, is that users must train the crawlers to match two criterions:

- *the crawler should find and return possibly all relevant information about a topic*; this can be done by covering all the necessary aspects of the topic, by adding all the necessary sub-components to the keyphrase map. Also, to match this criterion better, the user has to find many good starting URLs (at least one for all aspects), for which he can use the Weblog algorithm mentioned earlier.

- *the crawler should not return irrelevant documents*; this should be done by adjusting each component, which describe a single aspect, by adding the relevant keywords, and by further fine tuning the map by adding edges between the words. However, we must note the following. We believe that the way the words in a component are connected are only used for fine tuning when edges serve the purpose of disambiguating the meaning. In most cases, a set of words clearly describe a meaning. When this is not the case, for instance when three words can be contained in a sentence having two different meanings, the edges are also necessary to clarify the search criterion. Nevertheless, in this graph interpretation, the edges have an important role in the separation of components (aspects).

Note, that the present form of graph interpretation is similar to using boolean formulas to specify the search criterion (to be precise, this is equivalent to using formulas in disjunctive normal form), but this is done via a graphical interface, which is probably more easy to see through for a human than large boolean formulas.

Any kind of boolean formulas can also be used in keyword search engines like Google, however, it must be noted that the whole procedure differs from Google's search power in performance, since our formula matching only applies to the *extracted keyphrases of the documents*, while Google applies it to *any word* in the document.

We also modified the goal of the test runs, to give a different formulation for the testing of whether the usage of keyphrase maps is useful. Instead of having a test group and a control group with altered documents scores, the task was not to find as many documents as possible, but to train as good crawlers as possible.

The performance of the crawlers was measured by three measures derived from true/false positive and true/false negative documents. A document is positive if the crawler judges it to be a relevant document (based on its search criterion keyphrase map's similarity to the document's map). Thus a true positive document is one that was judged by the crawler to be a good document and is also judged as good by the classifier. A false positive is one judged to be relevant by the crawler, but not by the classifier. Similarly, a true negative is one judged to be irrelevant both by the crawler and the classifier, and a false negative is one thought to be irrelevant by the crawler, but relevant by the classifier. Using these concepts, the performance measures are as follows:

- $Accuracy = \frac{\#\ true\ positives}{\#\ all\ positives}$. This measures the fraction of truly good documents found by the crawler, thus optimizing this measure would yield the crawler not to return irrelevant documents
- $Recall = \frac{\#\ true\ positives}{\#\ true\ positives\ +\ \#\ false\ negatives}$. This measures the fraction of documents judged as good by the classifier returned by the crawler. Optimizing this would yield the crawler return all relevant information from an area
- $Crawling\ hit\ ratio = \frac{\#\ truly\ relevant\ documents\ found}{\#\ all\ documents\ processed}$. This measures how fruitful area the crawler is crawling. Optimizing this measure

would guide the crawler towards areas where many relevant documents can be found

Having these performance measures available for the user, the goal was to train crawlers with high performance measures. By examining how well the first two performance measures can be optimized, we expect to get an answer for how well the topic characterized by a document classifier can be approximated by a tool that is both understandable for humans and for the computer (since the usage of the SVM classifier as an interface between human and computer would be inappropriate, because of the SVM parameters being not meaningful for the human).

The two topics in these experiments were 'Natural language processing' and 'Cortical prosthetics'. Again, the testers were given 5 sample documents in the topic domain to start with.

5.5 Search strategies and experiences

After the experimental runs, the testers were interviewed about how they used the system, what strategies they devised, and what they found positive or negative.

Of course, the users started the search by extracting keyphrases from the sample documents. Then, these extracted keyphrase maps were 'cleaned', by means of deleting words that were thought to be not so much related to the topic in general. An initial user keyphrase map was created by putting together the previously created cleaned components into a unified map.

Also, to start crawlers, start URLs were also needed, since the sample documents were PDF documents, which could not be used as a starting point for the crawl. Thus, various combinations of the previously extracted keywords were entered to a Google query, to find potentially good starting URLs. The URLs marked by the SVM classifier to be a good document were usually used to start a crawl from, but occasionally non-accepted URLs with a high score were also experimented with.

Another simple strategy, introducing some human knowledge about the web proved to be fruitful: Usually, good PDF documents found by Google had a higher score, than good HTML pages, since their content is more well defined, they fit into the search topic more, especially when the search task is about a scientific domain. These documents were often contained in a list of other documents linked from an HTML page, such as a publication list of a research group, and usually, most documents in that list were quite similar, thus if one of them is a relevant one, than probably so are others. Thus, people started crawls from the home pages of the authors or the research group or institute. Often, they searched for the sample documents with Google, and started crawlers from URLs near that document, that they found promising.

When the crawlers found relevant documents, the keyphrases from them were also extracted, and the users' maps were updated taking those new phrases into account. One point in the task was to cover the topic sufficiently, that is to build such a map, that reflects all aspects of the search topic (to make the crawler find all relevant information). To reach this, the following was applied: when a crawler found a document that had a low estimated relevance based on the user's map, but the document classifier judged the document as a relevant, the keyphrases of that document were also added as a new component to the user's map, this way sharpening the search. In order to satisfy the other performance criterion, namely to make the crawler return only relevant documents were in most cases controlled by the minimum relevance threshold parameter of the crawler. Some users found a threshold that worked pretty well in most cases (around 0.6). Also, in some users' strategies, when a crawler found an irrelevant document, the keyphrases extracted from those documents were erased from the users' keyphrase map, to avoid the crawler returning such more documents. With the modified maps then, new crawlers were started, maybe from the same start URL as their ancestors, in order to sweep it again with the new search criterion in mind.

A general experience was that the crawlers performed well only at the beginning of the crawl, around the start URL. After that, they were likely to get away from good areas, where their performance slowly fell. In many cases, the structure of the sites were found to be tree-like, rarely were links pointing out to some other good region (like to the page of another researcher working on the same topic). This made the users force the crawlers stay within a site, in order not to let their performance fall. But this resulted in the crawler sweeping the same region again and again, and not finding any more new relevant documents. Only few crawlers were able to find new good regions through links to other internet domains.

We also checked whether the users found the documents we used to train the classifier, or whether they found completely new ones. The first topic (Language) was trained on documents from all over the web, from many authors. The users found some of these documents (around one fifth), and found many new ones (about 400). The second topic (Prosthetics) was trained on documents downloaded from two sites, having about 50 documents from one site, and 15 from the other. Some users have found the site that contained the 50 documents, and had crawlers that returned half of those documents. Also a few hundred new documents were found in this topic, but less then in the first one, this has proven to be a more difficult topic.

To see how well the users were able to identify the topic described by the SVM, we tested some maps that they have constructed. We launched crawlers with those maps to sweep the page where the most training documents were downloaded from for the second topic[3]. We tested four maps, all of them reaching a recall of 90-95% and an accuracy of 53-58%. We also verified the SVM on these documents, it had a performance of around 96%. This means that the maps were able to identify the topic quite accurately, the crawlers had returned almost all relevant information, and only less than the half of the returned documents were not from the topic that the SVM was trained for. It must be noted, that the other documents on that page were also from a similar domain, that is the task was quite hard.

There are some sites ideal for these crawling tasks: were (PDF) documents are virtually interlinked by HTML references, like CiteSeer, or arXiv, Rexa, etc. The advantage of these sites is that the *related documents* are strongly interconnected, virtually, clusters of documents related to a topic are formed. These connections could be easily exploited by the crawlers. However, most of these sites do not let robots make requests, thus the users were unable to utilize these sources.

One user also found it unnecessary and overcomplicated to use crawlers for internet searches. He admitted that he heavily relied on the keyphrase map extraction and map interaction properties of the system, but said that once the right keywords are known, Google-like search engines can also do the job for us. Even, when the keywords are not so well developed in the beginning of the search, resorting the results of Google based on keyphrase map similarity helps a lot. This is a valuable technique – provided that information is not changing quickly, and if the search task concerns the web – because the keyword map that identifies the decision surface can be constructed quickly. Nonetheless, there is a huge amount of context like information in the details of the crawl [Lőrincz et al., 2002, Kókai and Lőrincz, 2002] that can not be extracted by the method relying on Google alone.

We found that the 'news forager'-like crawling technique that we have used must be somewhat adopted when using it for topic specific, user controlled searching. First, it seems so, that the internet domains containing documents in similar topic are not well connected enough. Connections between topological internet domains is much dense, which makes the crawlers get lost easily. Second, when using crawlers that adopt to the search criterion, the crawlers need some time to tune their parameters, thus in the beginning of this learning period, they do not perform well. However, when a user starts a crawl having a narrow topic as a search criterion, the crawler has no time to get well trained, if short crawls are used, as it seems users would prefer. Starting many crawlers for various search tasks and waiting for the results for a few hours or even half a day did not seem to be a popular practise.

A possible solution could utilize crawlers that perform quick shallow (or adaptive) searches around areas which it finds to be promising (for example

[3]http://www.ninds.nih.gov/funding/research/npp/resources/archived_contracts.htm

a main page of a site), and would make longer hops across sites to look for new areas. Another possibility is to introduce communication between crawlers and to lessen the need for adaptivity but sharpen the evaluation of the context of a given neighborhood on the Internet. This latter has been studied [Lázár et al., 2006] and good results can be expected in the future.

Plots of example runs from the experiments that display the performances of the crawlers and some user keyphrase maps are listed in the Appendix.

Chapter 6

Appendix A: Software Users' Guide

6.1 Running the software

Running the software requires Java 1.5. The necessary *jar* files must be supplied with the *-classpath* parameter. The main class of the program is *nipg.crawlerengine.crawlerclient.RunCrawlerClient*. The command line arguments of the program are the following:

- **local/jxta:** whether to run the crawler server locally or remotely using JXTA technology. This must be the first parameter.
- **-shost "submission_host_name":** host name of the computer that runs the document submission server. Default value: **localhost**
- **-sport "submission_port":** server side port number used to connect the document submission server. Default value: **9090**
- **-scport "submission_client_port":** client side port number used to connect the document submission server. By default, a new port not yet in use is chosen automatically
- **-stimeout "submission_timeout":** timeout interval for document submission, in milliseconds. Default value: **10000**
- **-lhost "login_host_name":** host name of the computer that runs the login server. Default value: **localhost**
- **-lport "login_port":** port number used to connect the login server. Default value: **9091**
- **-lcport "login_client_port":** client side port number used to connect the login server. By default, a new port not yet in use is chosen automatically

- **-ltimeout "login_timeout":** timeout interval for login, in milliseconds. Default value: **3000**
- **-googlekey "google_key":** Google APIs registration key (needed for making Google queries through Google APIs). The default value is a presently registered, valid Google key, which may expire in the future. A new Google key can be obtained from `http://www.google.com/apis/` by creating a new account free of charge. It must be noted, that one key enables to issue 1000 queries per day, independently of which program issues them. For more queries, separate keys must be used.
- **-rdelay "refresh_delay":** the time interval for automatically refreshing the crawler output, in milliseconds (refreshing requires sending a query to the crawler server when not run locally). Default value: **3000** if the server is local, **30000** otherwise
- **-nologin:** do not use login. By default, users must log in before they can use the software. This is only for basic user identification

For example, to run the program int local mode, with the submission and login servers running on *nipglab06.inf.elte.hu*, type the following:
java -classpath dist*.jar nipg.crawlerengine.crawlerclient.RunCrawlerClient local -shost nipglab06.inf.elte.hu -lhost nipglab06.inf.elte.hu

6.2 Using the software

6.2.1 Login

After starting the software, the program loads the necessary data files, and the main GUI window is shown, with the login window popping up (if **-nologin** is not specified). Here, the user has to supply his login name and password (6.1). One can register a new username, where he has to supply his full name, a new user name, and a password (6.2).

After logging in, the main window can be seen. The window is split into three regions. The size of the regions can be varied by grabbing and moving the splitter lines with the mouse.

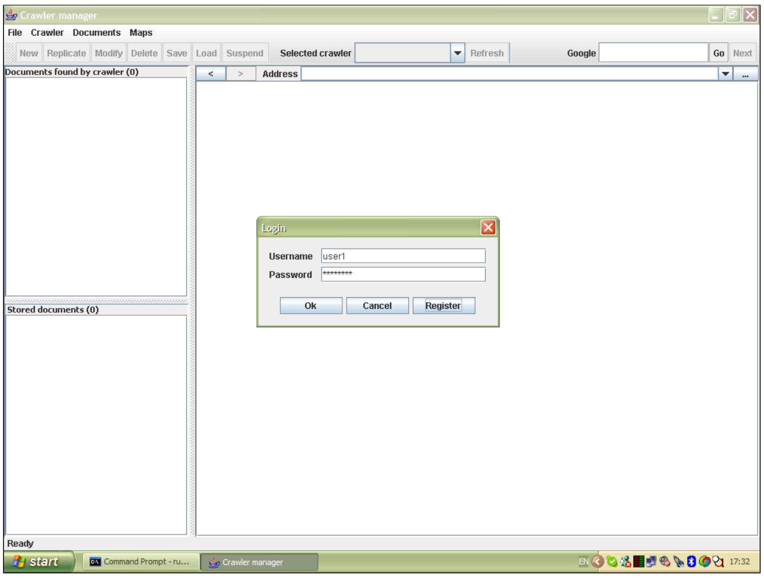


Figure 6.1: **Logging in to the system**
The user must supply his user name and password

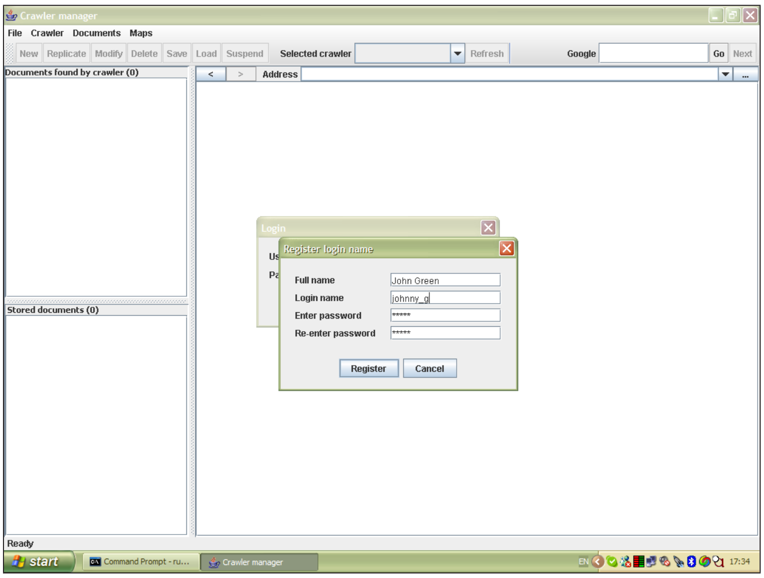


Figure 6.2: **Register a new login name**
The user must supply his full name, a user name that is not already in use, and a password

6.2.2 Browser area

The right panel is the browsing window, used to display pages and documents reached during the usage of the system. This also functions as an internet browser. One may enter an internet address into the text field on the top, and the page is loaded to the window after pressing enter. Alternatively, a previously entered address can be reloaded by selecting it from the drop-down list. The *previous* (<) and *next* (>) buttons can be used for traversing the browsing history. The button in the top right corner with the caption '...' can be used for loading a local file to the browser panel (6.3). One may also follow the links in HTML pages. If one opens a pdf file, the text contents of the file is extracted and converted to HTML, and then displayed (6.4).

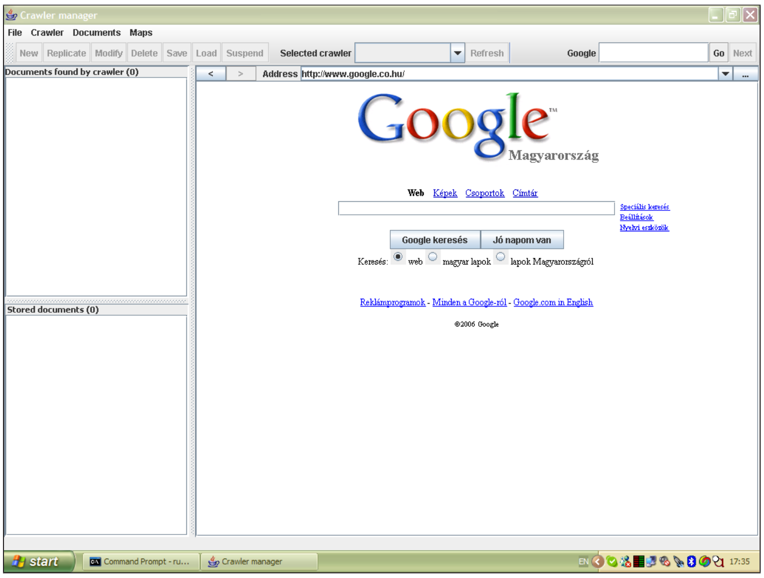


Figure 6.3: **Browser window**
The basic browsing functions like loading a page and following the links can be performed here

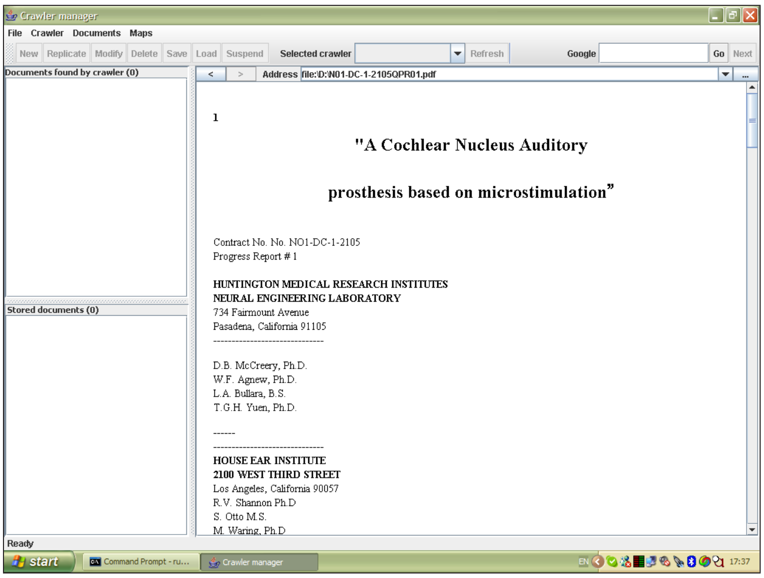


Figure 6.4: **Displaying a pdf file**
A text content of a pdf file is converted to HTML

6.2.3 Document lists

On the left side of the main split pane are two lists, that may contain document URLs. When one selects a document from the lists, the document is displayed in the browser area.

The top list contains results found by the system; either by the crawlers or by Google. The list always contains the documents of the actually selected crawler, or the last Google query. The bottom list contains the documents stored by the user for later use. If a document is stored, it is saved to the hard drive, and is loaded when the program is launched again.

The document lists have popup menus which can be activated by right clicking on the list. Most menu commands apply to the selected documents in the list. For example, one may store some documents by selecting them in the top list, and clicking 'Store documents' in the popup menu (6.5). The lists enable multiple selection. However, only one list may have selected entries at a time, namely the active one. When one of the two lists is activated (by performing some mouse action in them, like selecting an item), the other one is deactivated and selection is lost. The lists can also be sorted by various criteria, this can also be reached from the popup menu. The document related tasks can also be performed using the 'Documents' menu on the top of the GUI window. Like in the popup menus, tasks apply to selected documents in the active list.

Figure 6.5: **List popup menus**
The popup menus on the document lists enable several document related functions like storing a document for later use

When the mouse cursor is moved over an entry in either of the lists, the system displays additional information about the documents. Common information are the document index, document scores, and whether the document is a hit. The index is simply used to sort documents by arrival. The ‘score’, which is a real value between 0 and 1 tells the user how good the document is. The ‘hit’ flag indicates whether the document is accepted as a relevant one in the actual search topic. In addition, accepted documents are colored green in the list, while not accepted documents are colored red. Documents for which the evaluation is not yet available are colored black. The ‘reference score’ is also a goodness value, which is computed against a reference map, which can be set by the user, and serves as a helper in the search. For additional document information depending on how the document was found, see the forthcoming sections (6.2.5, 6.2.6).

The list of stored documents can also be saved to a file and loaded later, using either the popup menu, or the ‘File’ menu on the top of the main window. When a document is deleted from the list of stored documents, it is only deleted from the list, but not from the hard drive. It may still be contained in a previously saved list, or can be reloaded with the ‘Load all stored URLs’ menu item. Only when a document is erased with the ‘Erase’ menu item, then is it deleted from the hard drive.

6.2.4 Keyphrase maps

When a document is loaded, keyphrases can be extracted from it, by right clicking the document pane, and selecting 'Extract keyphrases' from the popup menu. The keyphrase map is displayed in a separate window (6.6). This window can be used for modifying the keyphrase map, as well as for saving it. At other points of the program, the saved maps can be loaded for various usage.

The keyphrase map can be edited by right clicking the map area, which brings up a popup menu. Right clicking on a blank area enables adding a new node, while right clicking an existing node enables modification. A new node may also be added by double (left) clicking the blank area. The popup menu enables adding, modifying, and deleting nodes, starting a new map, saving and loading maps, unifying maps by adding saved maps to the one being edited, extracting keyphrases from text files, and extending the map with synonyms of the selected node (6.7). Nodes may be selected and moved by selecting a region on the map with the mouse and dragging them. When adding a new node, a string and a weight must be supplied (6.8). The nodes are colored according to their weight. Zero weighted nodes have a white background, while nodes with a weight of one have an orange background, and the values in between are interpolated linearly. Two nodes can be connected with an edge, by right clicking in the middle of a node and dragging the cursor to the middle of an other node while holding down the right mouse button, where it can be released (6.9).

Keyphrase map editing can also be performed by opening a map editor window from the 'Maps | Map editor...' menu item found on the top of the main window. The 'Maps | Edit reference map...' menu item lets the user set a reference map, which can be used to make the system compute reference document scores, which are similarity values computed by comparing the document maps to this reference map.

Extending the map with synonyms requires a synonym database, and the performance heavily depends on it. By default, a general database is being used, extracted from the British National Corpus. Unfortunately this performs poorly on narrow scientific domains.

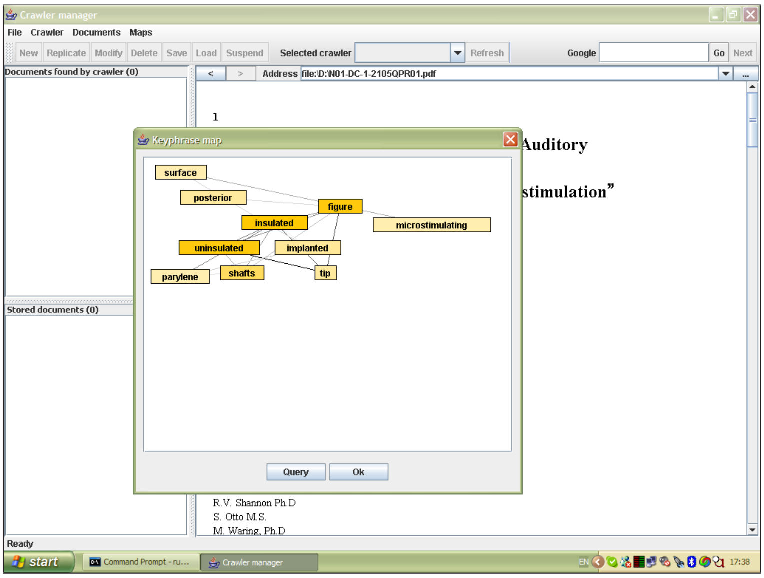


Figure 6.6: **Extracting keyphrases**
The keyphrase map is displayed as a word graph, which can be modified and saved for later usage

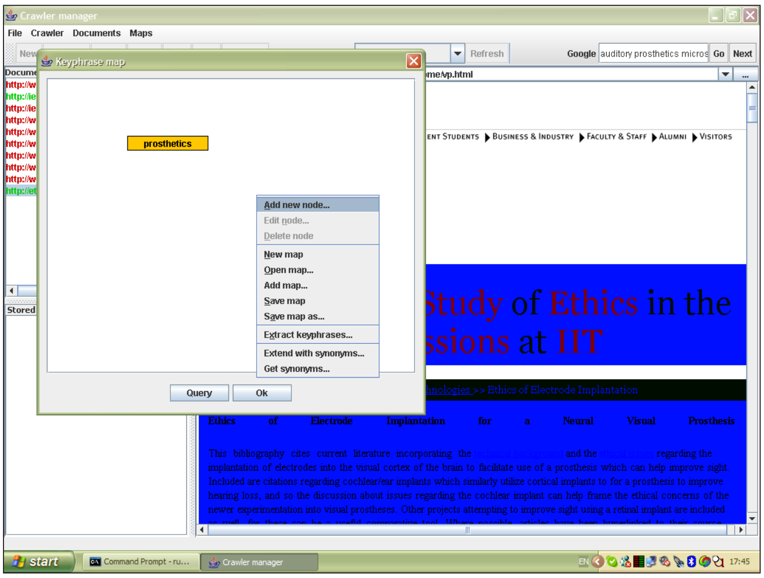


Figure 6.7: **Editing a keyphrase map: popup menu**
The popup menu offers several functions, like creating a new map, editing, saving, loading and extending maps with synonyms

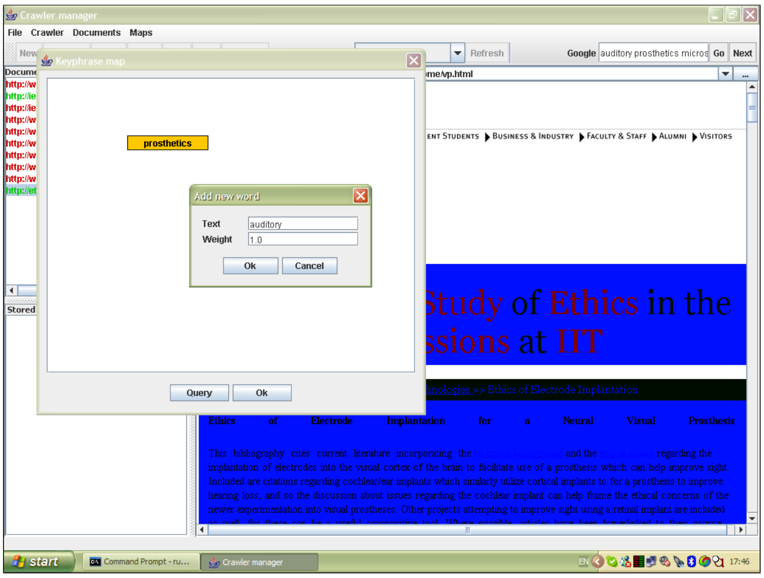


Figure 6.8: **Editing a keyphrase map: adding a new node**
To add a new node, a string and a weight must be supplied

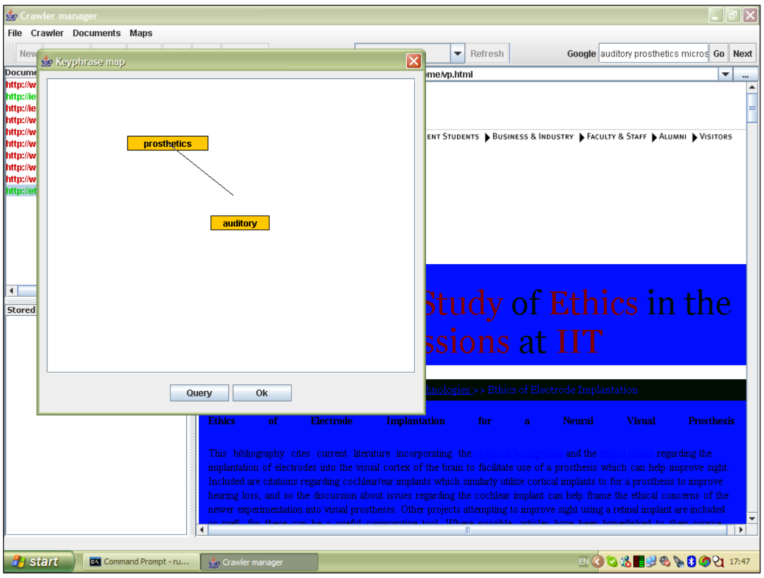


Figure 6.9: **Editing a keyphrase map: adding a new edge**
Edges between nodes can be inserted by dragging the mouse from the middle of one node to the middle of the other, while holding the right button down

6.2.5 Integrated Google interface

The user may perform keyword searches via the integrated Google interface, which is located on the right side of the toolbar, in the top right corner of the main window. In the text field, queries can be entered as one would usually do on the Google page. The query is performed either by pressing enter, or clicking the ‘Go’ button. Then, the first 10 results are returned and listed in the top list on the left. If the user clicks the ‘Next’ button, the next 10 results are also listed, and so on (6.10).

When the mouse cursor is moved over a document URL in the list, additional Google specific information is also displayed about the document: the title of the page, and the snippet extracted by Google which indicates in what context the keywords were found. These are the same information one would see when using Google from a browser. When an item is selected in the list, the page is loaded and displayed in the browser area (6.10).

When crawler related documents are displayed in the top left list, the results of the last Google query can be redisplayed in the list by clicking on the ‘Google’ label in the top right corner in front of the Google query text field.

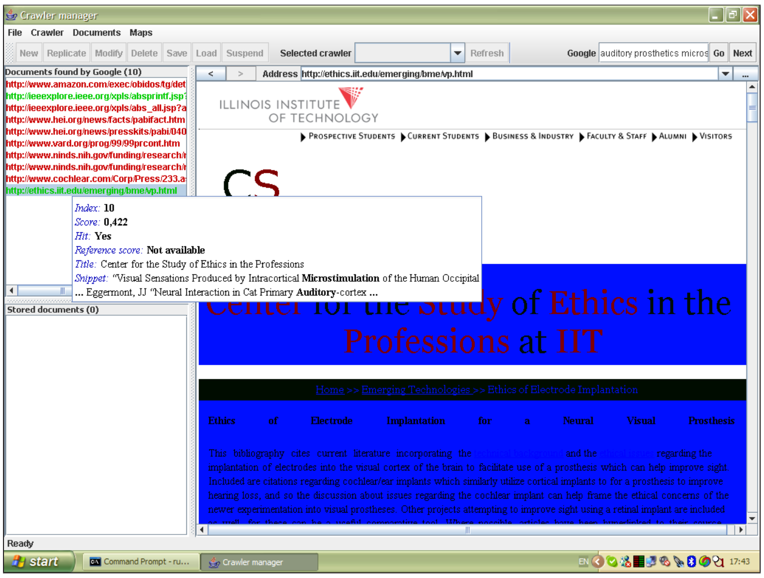


Figure 6.10: **Google query results**
The query results are displayed in the top left list. Additional information about the found documents can also be displayed

6.2.6 Crawlers

The left side of the toolbar on the top of the main window is devoted for crawler related operations. New crawlers can be created, existing crawlers can be modified, saved, deleted, loaded, suspended and resumed.

When creating new crawlers, the following parameters can be set:

- **unique name:** it is used for identifying the crawler in the system; must not be empty
- **start URL:** it defines where the crawl will begin from; must not be empty
- **site URL:** this tells the boundaries of the crawler, which it cannot leave. More precisely, it specifies which URLs are enabled for download. This works in a template matching fashion. The site URL does not need to be an existing URL, instead, the parts of the URL specified as site URL must be contained in the document URL to be enabled for download; that is the site URL's protocol must equal the document's protocol, the site URL's host name be must contained in the document's host name, the site URL's file path must be contained in the document's file path. For example, if the site URL is `http://cnn/2006/`, then the document's host must contain 'cnn' and the file path must contain '2006'. Thus, `http://www.cnn.com/2006/news.html` is accepted, `http://cnn.money.com/sports/2006/news.html` is also accepted, but `http://www.google.com/cnn/2006/news.html` is not accepted, since the document's host does not contain 'cnn'.
- **keyphrase map:** this specifies the search query. This map is matched by the crawler against the keyphrase maps extracted from the documents, and the ones with a match value exceeding a certain threshold will be returned to the user.
- **min relevance:** the above mentioned threshold, above which the documents are returned to the user; should be a real value between 0 and 1
- **path length:** the number of steps the crawler should make before it is restarted from a newly selected URL; should be greater than 1
- **URLs to visit repeatedly:** the crawler proceeds the following way: starting from a URL, it makes *path_length* number of steps, and then it selects a random URL from the list of URLs to visit repeatedly, and restarts crawling from that point. Initially, such URLs can be supplied manually. As the crawler proceeds, it collects potentially good starting URLs, which it stores into a list, along with a value. This list can be

queried by the user, when it modifies a crawler. When starting a new crawler, the list of an existing crawler can be imported to the crawler going to be created. Editing the list can be done by right clicking it, and selecting the options in the popup menu.

- **expect feedback:** whether to expect manual feedback from the user for the documents found. When checked, the user may (should) send feedback (good/bad) to the crawler for each document using the document list popup menu. When not checked, feedback is sent automatically to the crawler, based on the 'hit' flag of the documents

While any of the parameters have a wrong value, the create button cannot be pressed. Wrong parameter strings are indicated by red printing color.

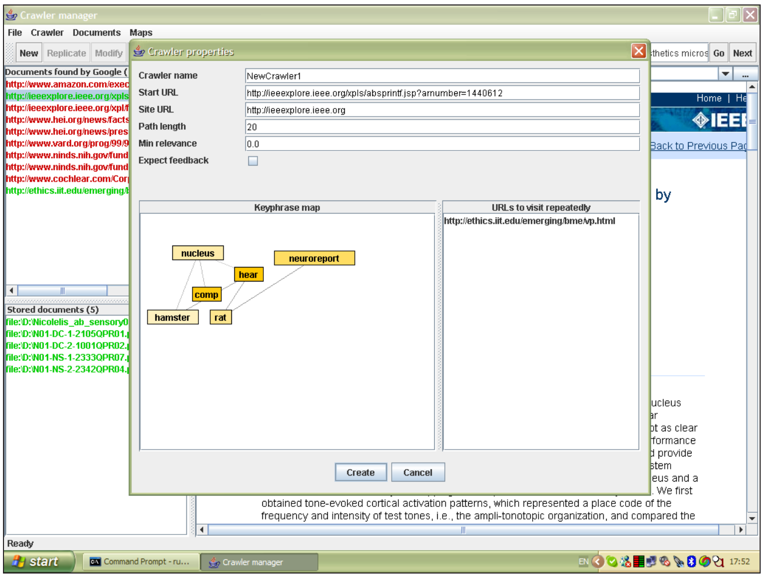


Figure 6.11: **Creating a crawler**
To create a crawler, a name, a start URL, a site URL and a keyphrase map specifying the search query must be supplied

After the crawler is created, it is inserted into the list of currently running crawlers. The crawler, whose output is currently displayed can be selected from the drop down list in the middle of the crawler toolbar. The output of the selected crawler can be refreshed by clicking the 'Refresh' button next to the crawler selection drop down list. Refresh is also performed automatically, the refresh interval can be set by the *-rdelay* parameter of the program. The selected crawler can be saved, and a previously saved crawler can be loaded. The crawler can be suspended, which means that it pauses running until it is resumed. However, it may also happen that a crawler is automatically suspended by the system, when it has found more than 100 documents that have not yet been seen (and stored or deleted) by the user. A crawler can also be replicated, which means starting a new crawler with parameters similar to the selected one (the crawler creation window is popped up, filled with the parameters of the selected crawler, and the parameters can be modified). Crawler related actions can also be performed using the 'Crawler' menu on the top of the main window.

The documents found by a crawler are displayed in the top left list. Additional information about the documents is the estimated 'relevance', which is a similarity value of the document's keyphrase map compared to the maps of the sample documents specifying the actual search topic. When a document is selected from the list, it is displayed in the browser area.

The status of the crawler is displayed in the bottom of the window, in the status bar. The 'Active' check box indicates whether the crawler has stopped running (which can only happen if it does not find any links to follow, and it does not have any URLs to restart from). The 'Suspended' check box indicates whether the crawler is suspended or not. The other fields display the performance scores of the crawler. 'Accuracy' tells the ratio of relevant to irrelevant documents returned by the crawler to the user; that is how well the crawler can filter out unwanted documents. 'Recall' tells the ratio of returned relevant documents to all relevant documents (i.e. also those relevant documents that were passed by by the crawler, but not returned to the user); that is how well the crawler is able to recognize relevant information. 'Crawling hit ratio' means the ratio of relevant documents to all documents passed by the crawler; that is how good a region the crawler is sweeping. By right clicking on these labels, it can be set whether the recent statistics (facts calculated from the last 100 documents passed by by the crawler) or the overall statistics should be displayed.

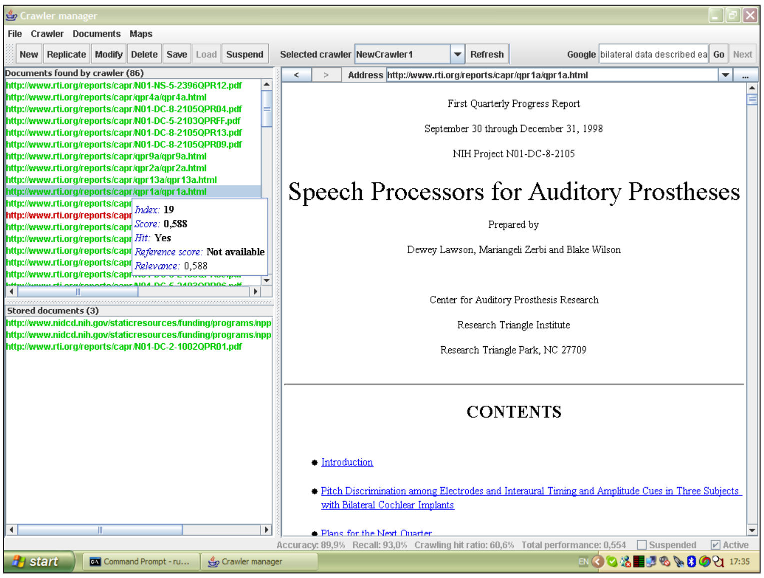


Figure 6.12: **Documents found by a crawler**
The documents found by a crawler are displayed in the top left list. Additional information about the found documents can also be displayed

Chapter 7

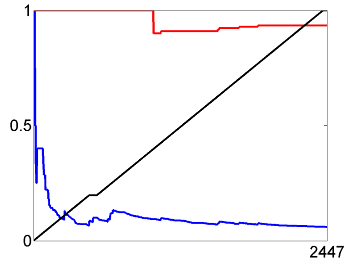
Appendix B: Test examples

The following figures are examples from the tests that we have performed. The first group of figures ((7.1, 7.2, 7.3)) show example crawler runs, the changing of the Accuracy (blue) and Recall (red) performance measures. When Accuracy is not changing, it indicates that the crawlers had found true or false negative documents; when recall is not changing, it indicates that it had found false positive or true negative documents. When both accuracy and recall are constant, it indicates that the crawler had encountered true negative documents. Also, another interesting value is displayed (black), the ratio of new documents to all documents, which shows when the crawlers encountered areas with new information (i.e when the black line goes high up), and when they have swept already seen areas again (i.e. when the black line goes horizontally).

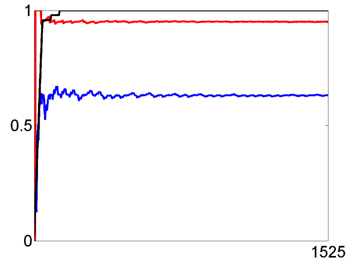
Some of these plots display runs that had very good recall but poor accuracy, some that were poor at the beginning but became better at the end, some that kept very good because the crawler was forced to stay in a good region, some where the performances dropped at the end because the crawler went to a poor region further from the start URL, and many runs where the performance stayed around 50% for long which is quite a good performance from a search system (it finds 50% of the relevant documents, and 50% of the returned documents are truly relevant).

We have also plotted these figures with the true negative documents not included, since they are irrelevant for the performance measures (7.4, 7.5, 7.6).

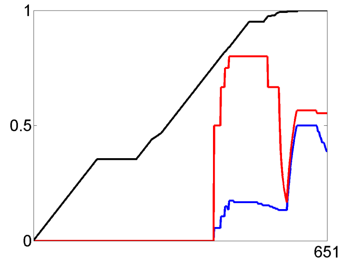
The other group of figures show example maps that users had created for the crawlers, two from both topics (7.7, 7.8, 7.9, 7.10). These were chosen from the maps belonging to very fruitful runs.



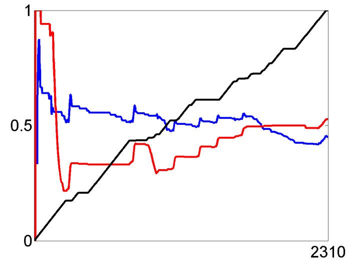
(a) Sample run 1



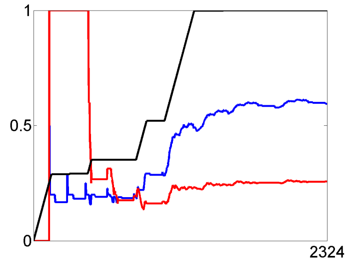
(b) Sample run 2



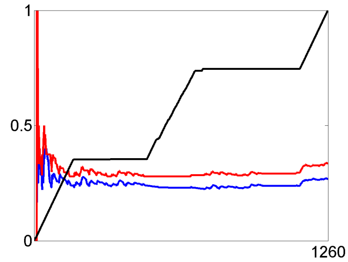
(c) Sample run 3



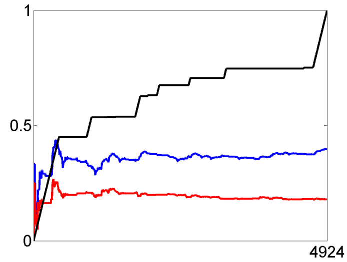
(d) Sample run 4



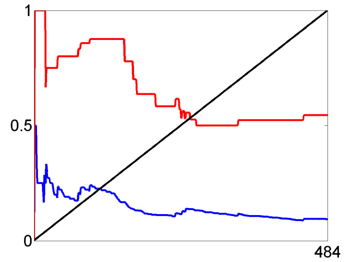
(e) Sample run 5



(f) Sample run 6

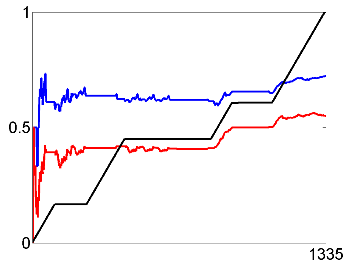


(g) Sample run 7

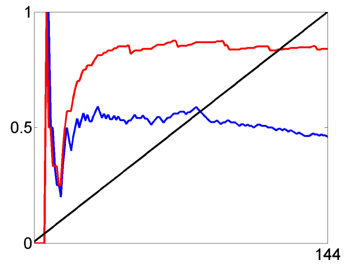


(h) Sample run 8

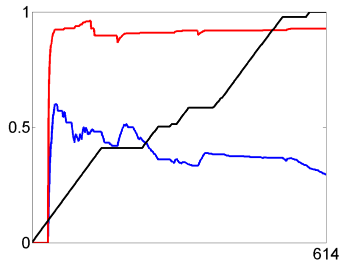
Figure 7.1: **Sample runs**
Blue: Accuracy, red: Recall, black: the ratio of new documents to all documents. The horizontal scale displays the documents encountered by the crawler, the total number of documents is shown in the bottom right corner



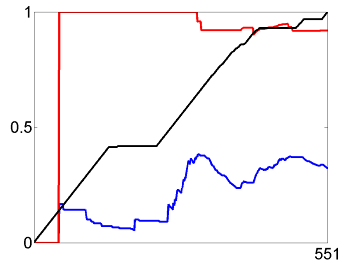
(a) Sample run 9



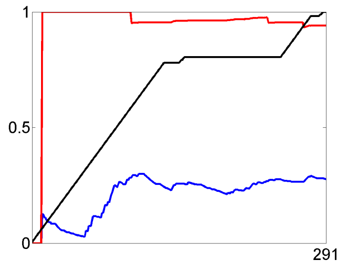
(b) Sample run 10



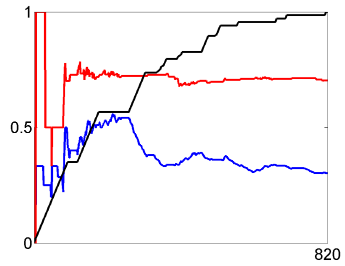
(c) Sample run 11



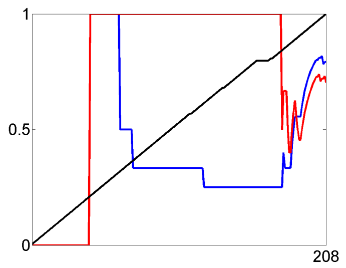
(d) Sample run 12



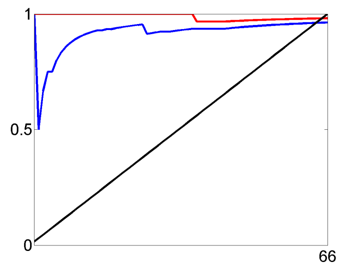
(e) Sample run 13



(f) Sample run 14

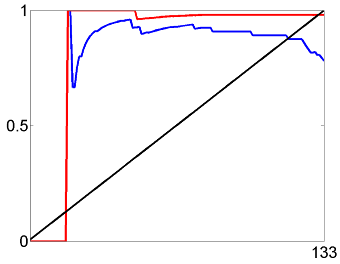


(g) Sample run 15

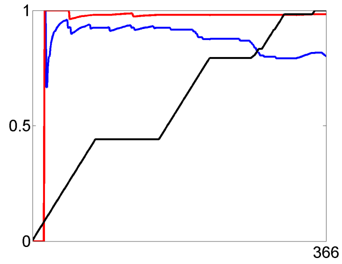


(h) Sample run 16

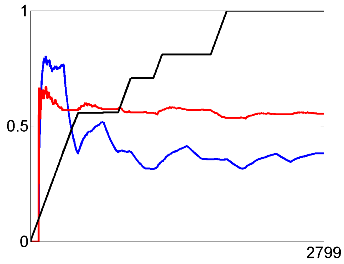
Figure 7.2: **Sample runs**
Blue: Accuracy, red: Recall, black: the ratio of new documents to all documents. The horizontal scale displays the documents encountered by the crawler, the total number of documents is shown in the bottom right corner



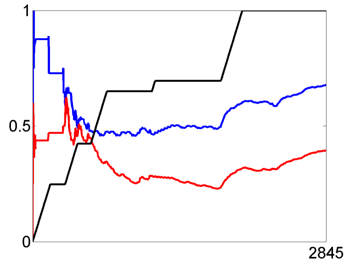
(a) Sample run 17



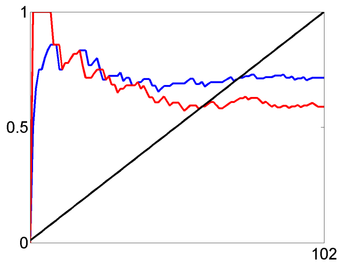
(b) Sample run 18



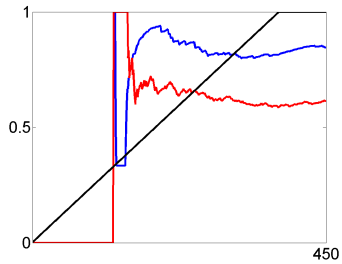
(c) Sample run 19



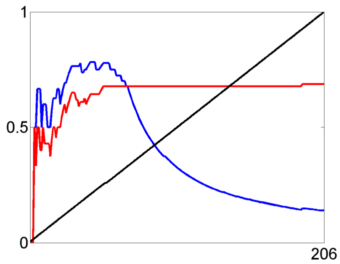
(d) Sample run 20



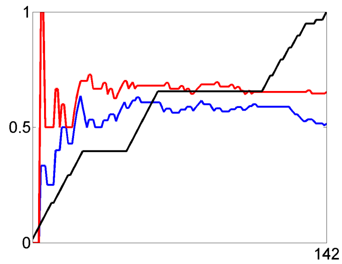
(e) Sample run 21



(f) Sample run 22

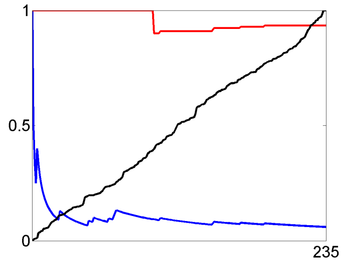


(g) Sample run 23

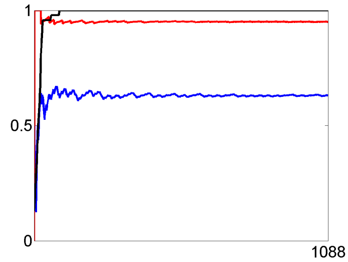


(h) Sample run 24

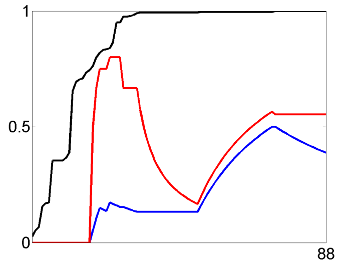
Figure 7.3: **Sample runs**
Blue: Accuracy, red: Recall, black: the ratio of new documents to all documents. The horizontal scale displays the documents encountered by the crawler, the total number of documents is shown in the bottom right corner



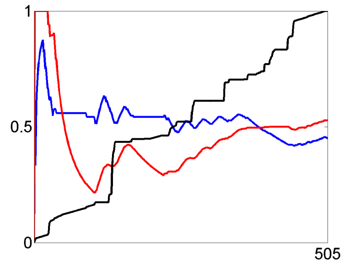
(a) Sample run 1



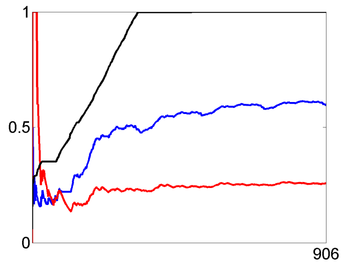
(b) Sample run 2



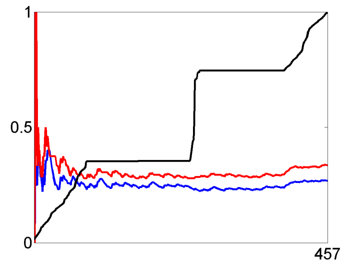
(c) Sample run 3



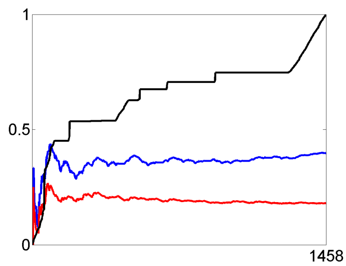
(d) Sample run 4



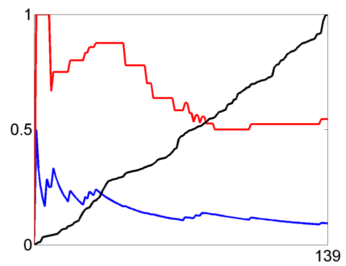
(e) Sample run 5



(f) Sample run 6

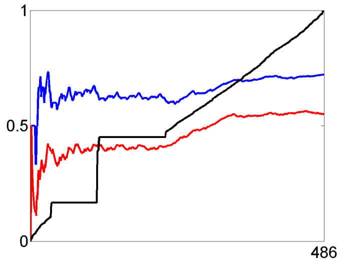


(g) Sample run 7

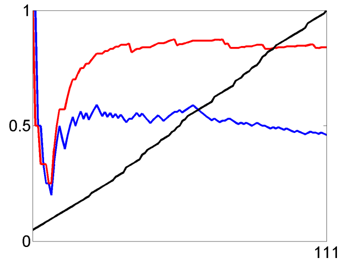


(h) Sample run 8

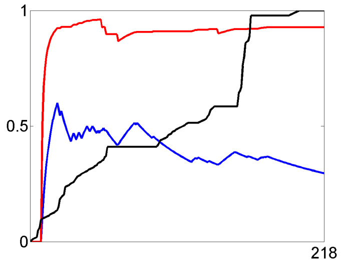
Figure 7.4: **Sample runs**
Blue: Accuracy, red: Recall, black: the ratio of new documents to all documents. The horizontal scale displays the documents encountered by the crawler, the total number of documents is shown in the bottom right corner. The true negative documents are left out from the series



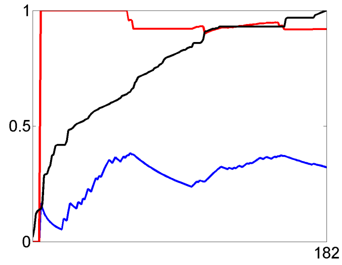
(a) Sample run 9



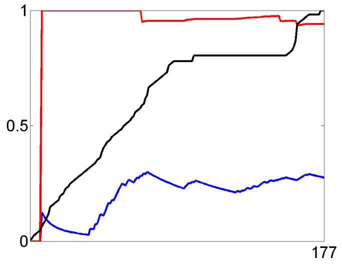
(b) Sample run 10



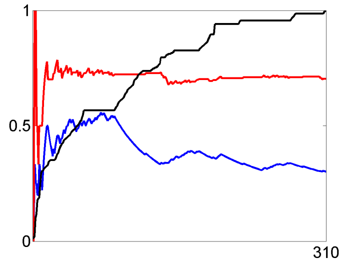
(c) Sample run 11



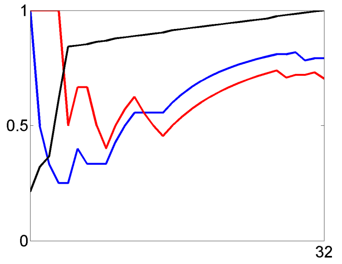
(d) Sample run 12



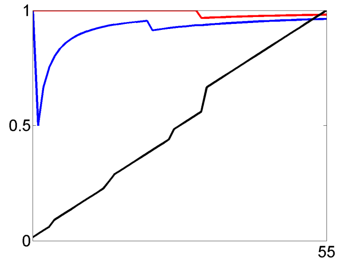
(e) Sample run 13



(f) Sample run 14

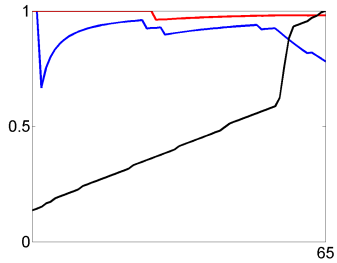


(g) Sample run 15

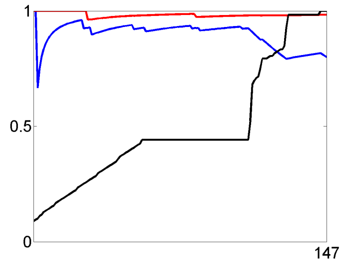


(h) Sample run 16

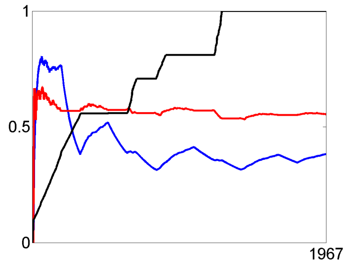
Figure 7.5: **Sample runs**
Blue: Accuracy, red: Recall, black: the ratio of new documents to all documents. The horizontal scale displays the documents encountered by the crawler, the total number of documents is shown in the bottom right corner. The true negative documents are left out from the series



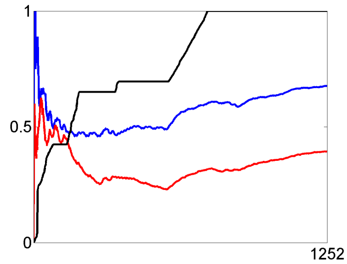
(a) Sample run 17



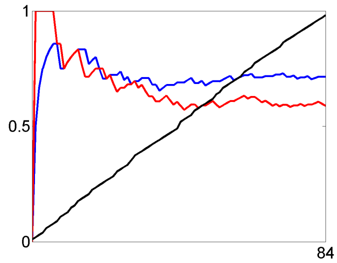
(b) Sample run 18



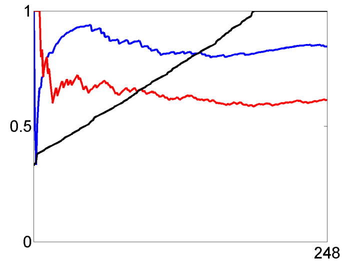
(c) Sample run 19



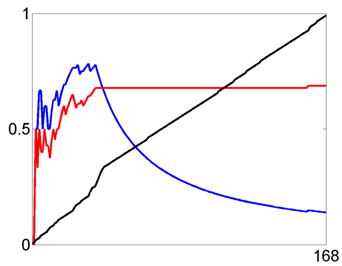
(d) Sample run 20



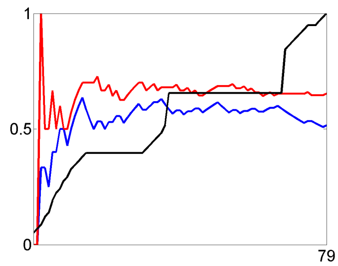
(e) Sample run 21



(f) Sample run 22



(g) Sample run 23



(h) Sample run 24

Figure 7.6: **Sample runs**
Blue: Accuracy, red: Recall, black: the ratio of new documents to all documents. The horizontal scale displays the documents encountered by the crawler, the total number of documents is shown in the bottom right corner. The true negative documents are left out from the series

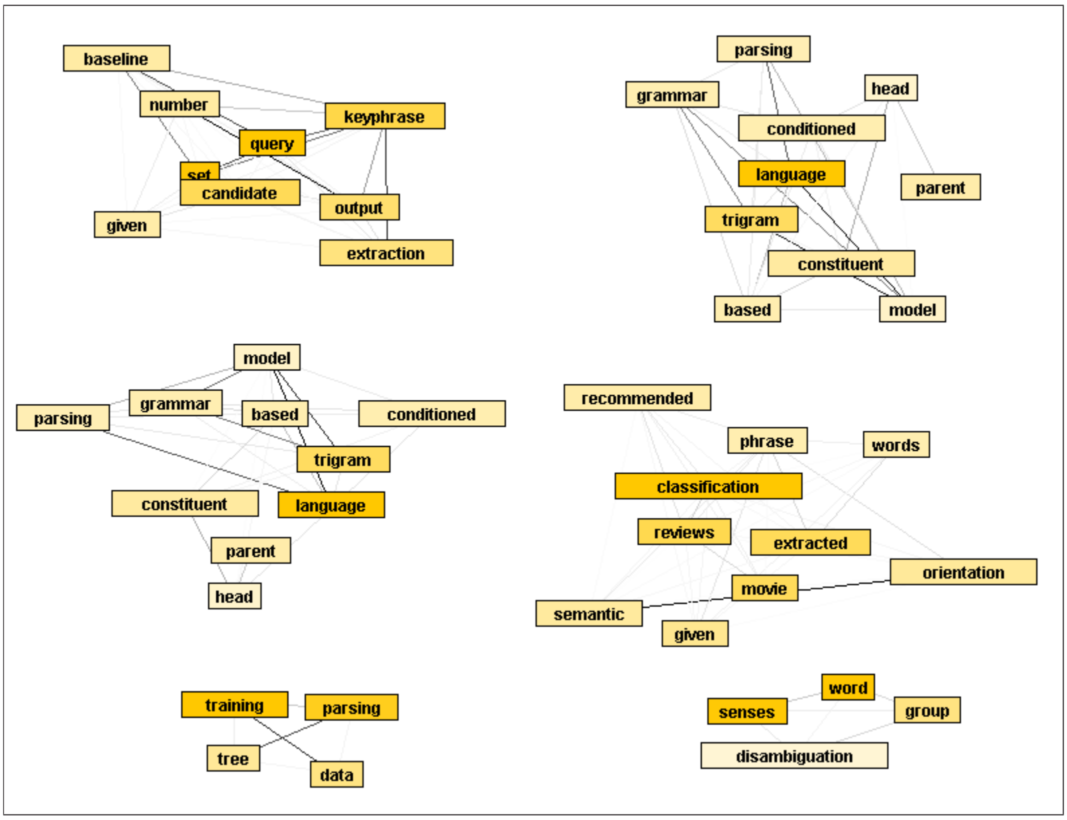


Figure 7.7: **Sample map 1 from topic 'Language'**

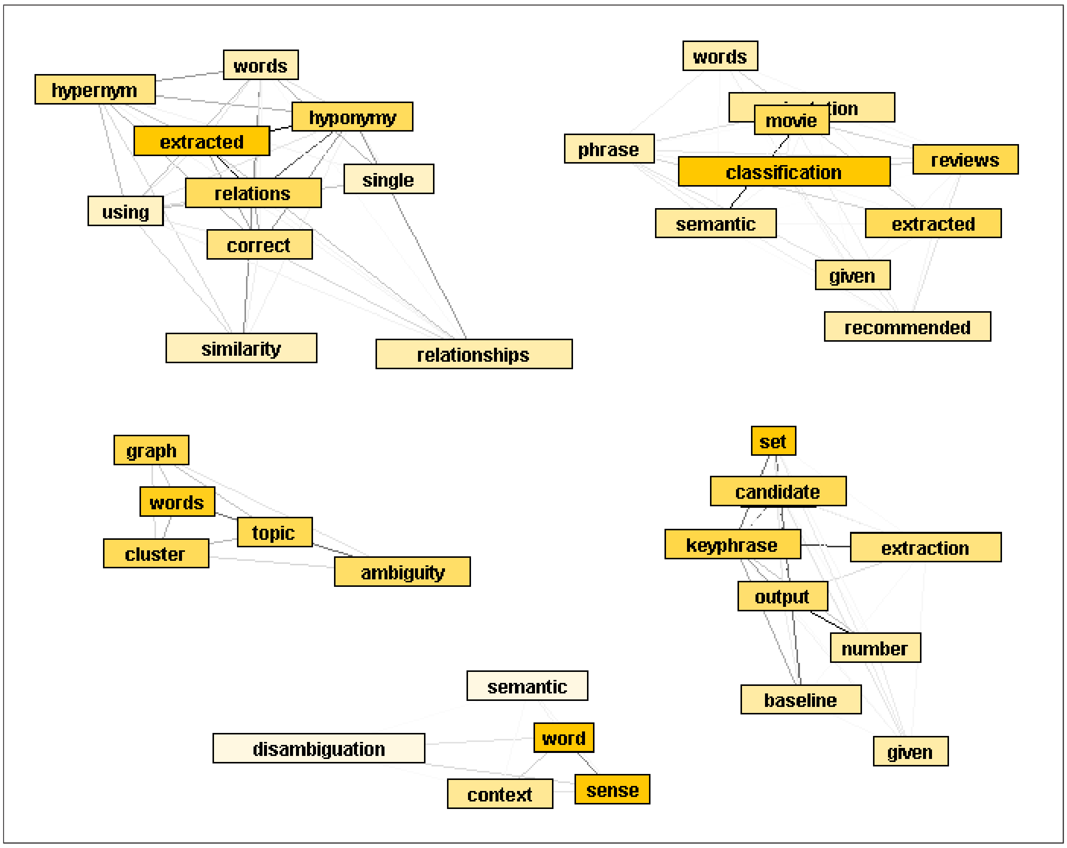


Figure 7.8: **Sample map 2 from topic 'Language'**

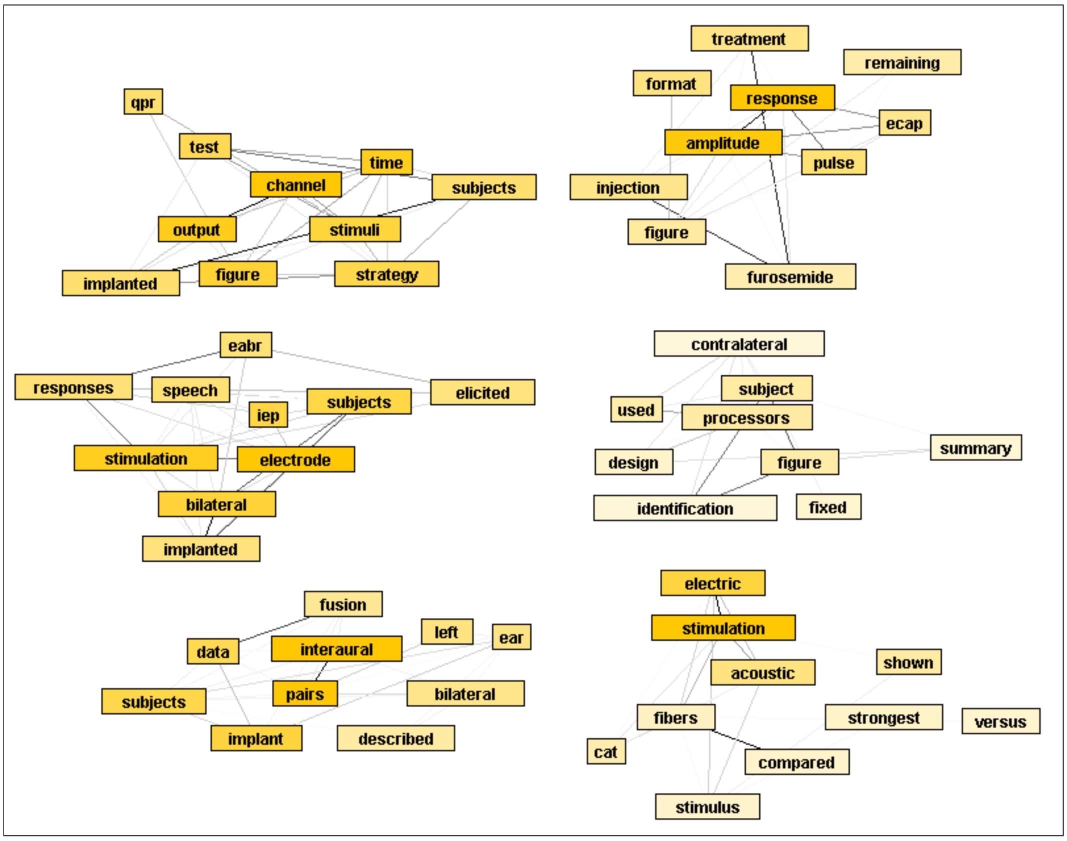


Figure 7.9: **Sample map 1 from topic 'Prosthetics'**

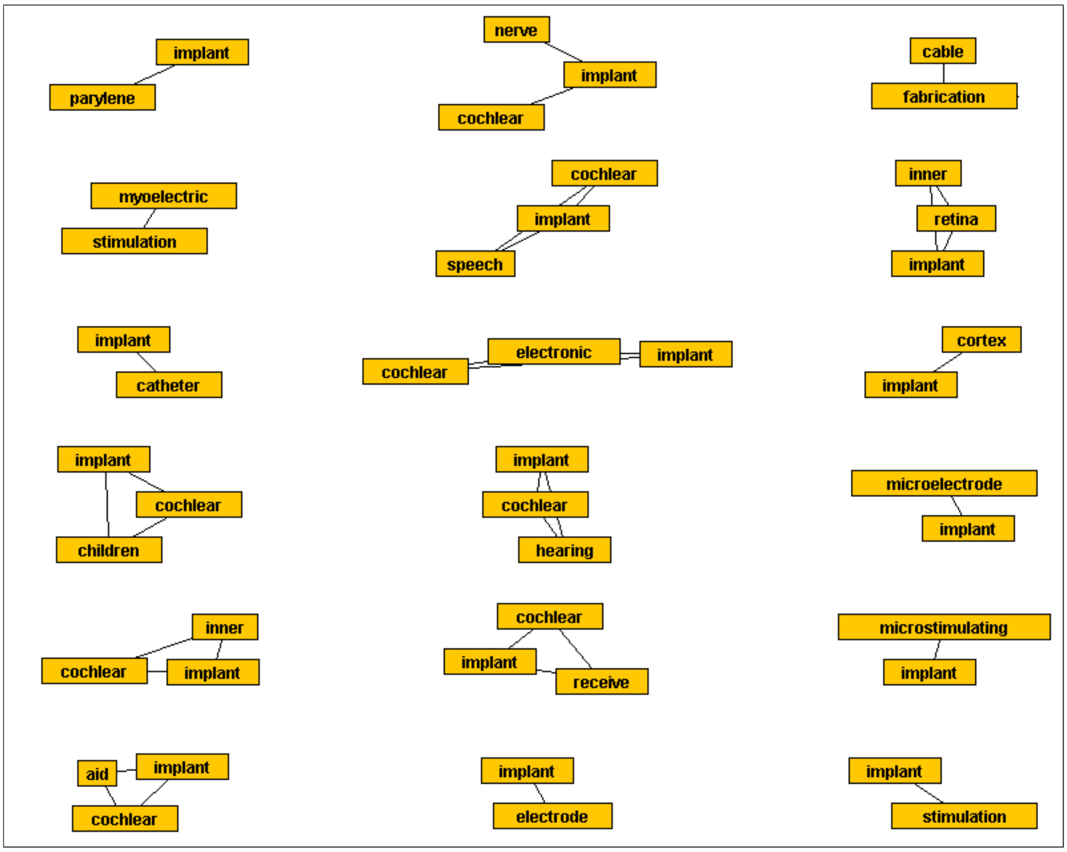


Figure 7.10: **Sample map 2 from topic 'Prosthetics'**

Bibliography

[Biederman, 1987] Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychol. Rev.*, 94:115–147.

[Frank et al., 1999] Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C., and Nevill-Manning, C. G. (1999). Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 668–673.

[Gábor et al., 2005] Gábor, B., Gyenes, V., and Lőrincz, A. (2005). A corpus-based neural net method for explaining unknown words by wordnet senses. In *Lecture Notes in Artificial Intelligence 3721*, pages 470–477, Berlin. Springer-Verlag.

[Harnad, 2003] Harnad, S. (2003). Cognition is categorization. UQAM Summer Institute in Cognitive Sciences on Categorization, June 30 - July 11, 2003, http://www.unites.uqam.ca/sccog/liens/program.html.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554.

[Kókai and Lőrincz, 2002] Kókai, I. and Lőrincz, A. (2002). Fast adapting value estimation based hybrid architecture for searching the world-wide web. *Applied Soft Computing*, 2:11–23.

[Lázár et al., 2006] Lázár, K., Palotai, Z., and Lőrincz, A. (2006). Efficiency of communication between internet crawlers in different worlds. manuscript in preparation.

[Lőrincz, 2004] Lőrincz, A. (2004). Intelligent encoding and economical communication in the visual stream, early cognitive vision workshop, workshop on coding of visual information in the brain, june 1, 2004, isle of skye, scotland, collection of abstracts. http://www.cn.stir.ac.uk/ecovision-ws/pdf/71.pdf, see, also http://arxiv.org/abs/q-bio.NC/0403022.

[Lőrincz et al., 2004a] Lőrincz, A., Gábor, B., Mandusitz, S., and Palotai, Z. (2004a). Bottom-up clustering and top-down shattering of scale-free environments for information fusion. Conf. on Information Fusion, Stockholm, Sweden.

[Lőrincz et al., 2004b] Lőrincz, A., Gábor, B., Mandusitz, S., and Palotai, Z. (2004b). Bottom-up clustering and top-down shattering of scale-free environments for information fusion, information fusion 2004, june 28 - july 1, 2004, stockholm, sweden. http://www.fusion2004.foi.se/papers/IF04-0471.pdf, see, also http://www.afosr.af.mil/pages/AFOSRWorkshop.htm.

[Lőrincz et al., 2002] Lőrincz, A., Kókai, I., and Meretei, A. (2002). Intelligent high-performance crawlers used to reveal topic-specific structure of the WWW. *International Journal of Foundations of Computer Science*, 13:477–495.

[Lőrincz and Szirtes, 2003] Lőrincz, A. and Szirtes, G. (2003). Towards a theory of consciousness: Proposal for the resolution of the homunculus fallacy with predictions. http://arxiv.org/abs/nlin.AO/0303042.

[Ferrer i Cancho and Sole, 2001] Ferrer i Cancho, R. and Sole, R. V. (2001). The small world of human language. *Proc. R. Soc. Lond. B*, 268:2261–2265.

[Palotai et al., 2005a] Palotai, Z., Farkas, C., and Lőrincz, A. (2005a). Selection in scale-free small world. In *CEEMAS*, number 3690 in Lecture Notes in Artificial Intelligence, pages 579–582, Berlin, Germany. Springer.

[Palotai et al., 2006a] Palotai, Z., Farkas, C., and Lőrincz, A. (2006a). Is selection optimal for scale-free small worlds? *Complexus*. in press.

[Palotai et al., 2005b] Palotai, Z., Gb́or, B., and Lőrincz, A. (2005b). Adaptive highlighting of links to assist surfing on the internet. *International Journal of Information Technology and Decision Making*, 4:117–139.

[Palotai et al., 2006b] Palotai, Z., Mandusitz, S., and Lőrincz, A. (2006b). Computer study of the evolution of 'news foragers' on the internet. In Abraham, A., Grosan, C., and Ramos, V., editors, *Swarm Intelligence and Data Mining*, Springer SCI Series. Springer, Berlin, Germany.

[Póczos and Lőrincz, 2005a] Póczos, B. and Lőrincz, A. (2005a). Independent subspace analysis on innovations. In et al., J. G., editor, *Proc. of the European Conf. on Machine Learning*, volume 3720 of *LNAI*, pages 698–706, New York, N.Y. Springer.

[Póczos and Lőrincz, 2005b] Póczos, B. and Lőrincz, A. (2005b). Independent subspace analysis using geodesic spanning trees. In *Proc. of the Int. Conf. on Machine Learning*. ICML 2005, Bonn (accepted).

[Póczos and Lőrincz, 2006] Póczos, B. and Lőrincz, A. (2006). Non-combinatorial estimation of independent autoregressive sources. *Neurocomputing*. in press.

[Shkarin, 2002] Shkarin, D. (2002). Ppm: one step to praticality. In *Proc. IEEE Data Compression Conf. (DCC'2002)*, pages 202–211.

[Steyvers and Tenenbaum, 2005] Steyvers, M. and Tenenbaum, J. B. (2005). The large-scale structure of semantic networks: statistical analyses and a model of semantic growth. *Cognitive Science*, 29:41–78.

[Turney, 1999] Turney, P. D. (1999). Learning algorithms for keyphrase extraction. *Information Retrieval*.

[Turney, 2002] Turney, P. D. (2002). Coherent keyphrase extraction via web mining. Technical report, Institute for Information Technology, National Research Council of Canada, M-50 Montreal Road, Ottawa, Ontario, Canada, K1A 0R6.